Development of an autonomous control system for target-optimised use of intralogistics means of transport in a production system for individualised products

by
Julian Grosse Erdmann

Thesis presented in fulfilment of the requirements for the degree of
Master of Engineering (Engineering Management)
in the Faculty of Industrial Engineering at Stellenbosch University
This thesis has also been presented at Reutlingen University, Germany, in terms of a double-degree agreement

Supervisor: Mr. Konrad von Leipzig
Co-supervisor: Prof. Dr.-Ing. Vera Hummel

March 2020

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2020

Abstract

Rapidly changing market conditions and global competition result in higher expectations from customers, and in turn, require increased efficiencies from companies. This, coupled with the increasing complexity of logistics systems, requires innovative approaches concerning the organisation and control of these logistics systems. In scientific research, concepts of autonomously controlled logistics systems show a promising approach to meeting the increasing requirements for flexible and efficient order processing.

In this context, this work aims to develop a system that is able to dynamically adjust order processing, and optimise intralogistics transportation with regard to various generic intralogistics target criteria in a flexible flow production. In this paper, the logistics system under consideration consists of various means of transport for autonomous decision-making and fulfilment of transport orders with defined source-sink relationships. The framework of this work is set by the development of a conceptual understanding of autonomous control and optimisation of several target figures in intralogistics. The two main target figures are costs and performance.

The core idea of the system's logic is to solve the problem of an order allocation to a specific means of transport by linking a Genetic Algorithm with a Multi-agent System. The Genetic Algorithm provides a global optimised solution to the problem, which is partially evaluated by a Multi-agent System, and then optimised based on local knowledge by monitoring and adjusting the appropriate decision variables in terms of problem-specific criteria. The developed model is based on the existing production system at the Werk 150, the factory of the ESB Business School on the Reutlingen University campus.

The behaviour of the system is first examined with the help of a simulation study. The results obtained from the simulation are tested with common verification and validation techniques in production and logistics to confirm the credibility of the system.

The work shows that the developed system leads to a higher logistical target achievement than conventional central planning and control concepts.

Keywords: intralogistics, optimisation, means of transport, autonomous control system

Opsomming

Vinnig veranderende marktoestande en wêreldwye kompetisie lei tot hoër verwagtinge van kliënte, en dus verg meer doeltreffendheid van ondernemings. Dit, tesame met die toenemende kompleksiteit van logistieke stelsels, verg innoverende benaderings rakende die organisering en beheer van hierdie logistieke stelsels. In wetenskaplike navorsing word konsepte getoon van outonome beheerde logistieke stelsels met 'n belowende benadering om aan die toenemende vereistes vir buigsame en doeltreffende bestelling verwerking te voldoen.

Die doel van hierdie werk is om 'n stelsel te ontwikkel wat in staat is om bestelling verwerking dinamies aan te pas en om intra-logistiese vervoer te optimeer met betrekking tot verskillende generiese intra-logistiese teiken kriteria in 'n buigsame vloei produksie. In hierdie navorsing word ‘n logistieke stelsel oorweeg wat bestaan uit verskillende vervoermiddele vir outonome besluitneming en die uitvoering van vervoer bestellings met gedefinieerde afhaal en aflewerings verhoudinge. Die raamwerk van hierdie werk word bepaal deur die ontwikkeling van 'n konseptuele begrip van outonome beheer en die optimalisering van verskillende teiken syfers in intra-logistiek. Die twee belangrikste teiken syfers is koste en prestasie.

Die kerngedagte agter die logika van die stelsel is om die probleem van 'n bestellings toewysing aan 'n spesifieke vervoermiddel op te los deur 'n genetiese algoritme met 'n “Multi-agent” stelsel te koppel. Die genetiese algoritme bied 'n wêreldwye geoptimaliseerde oplossing vir die probleem aan. Dit word gedeeltelik deur 'n “Multi-agent” stelsel geëvalueer en daarna geoptimaliseer op grond van plaaslike kennis deur die toepaslike besluitnemings veranderlikes te monitor en aan te pas in terme van probleem spesifieke kriteria. Die ontwikkelde model is gebaseer op die bestaande produksie stelsel van die Werk 150 van Reutlingen Universiteit.

Die gedrag van die stelsel word eers ondersoek met behulp van ‘n gesimuleerde studie. Die resultate van die simulasie word getoets met algemene bevestiging- en validasie tegnieke in produksie en logistiek om die geloofwaardigheid van die stelsel te bevestig.

Hierdie navorsing toon aan dat die ontwikkelde stelsel lei tot 'n hoër logistiese doelwitbereikings vlak teenoor huidige konvensionele sentrale beplanning en beheer konsepte.

Sleutelwoorde: intra-logistiek, optimalisering, verskillende vervoermiddele, outonome beheerstelsel

Acknowledgements

I would like to acknowledge and to thank the following people for their various contributions towards the completion of this work.

Many thanks to Prof. Dr.-Ing. Vera Hummel and Mr. Konrad von Leipzig as my supervisors and mentors of this work over the past two years. For their continuous support, helpful suggestions and constructive criticism as to its content and practical guidance, I am exceptionally grateful.

I am very grateful to all employees of the Werk150 for sharing with me their knowledge and experience. Especially to Jan Schuhmacher who answered my queries for this thesis, provided me with documentary material and helpful thoughts and contribution. I would also like to thank Mr. Vinu Vijayakumaran Nair for his patience and support during the implementation and validation phase.

Finally, I would like to thank my family and my friends for their emotional support and encouragement during the past two years.

Table of Contents

List of Figures

List of Tables

List of Acronyms

Abbreviations listed in the Oxford English Dictionary are not explicitly explained. Some of the listed German abbreviations denote proper names of organisations, which is why they have been used in English in their original format and not translated. Abbreviations that are only used in a figure or table and are explained in the same place are not explicitly listed here.

ACO	Ant colony algorithm
GA	Genetic algorithm
GRASP	Greedy randomised adaptive search procedure
JSON	JavaScript Object Notation
MES	Manufacturing Execution System
MTM	Methods-time measurement
OS	Order Sequence
PSO	Particle swarm optimisation
SA	Simulated annealing
SES	Self Execution System
TS	Tabu search
TSP	Travelling salesman problem
UML	Unified modelling language
V&V	Verification and validation
VRP	Vehicle routing problem

List of Symbols

DD_C	desired delivery times of the customer
DP_{total}	time for ordering the defective or missing part
$ER_{GI, m}$	total transportation time to/from the goods issue of means of transport m without transport order
$ER_{GI, m}$	total transportation time to/from the goods issue of means of transport m without transport order
$ER_{GI, x}$	transportation time to the goods issue without order
$ER_{GI, y}$	transportation time from the goods issue to the goods receipt
ER_{KR}	ratio of empty runs to utilisation time of the KollRo
ER_m	ratio of empty runs to utilisation time of all means of transport
$ER_{PO, m}$	total transportation time to/from the workstation picking order 1/2 of means of transport m without transport order
$ER_{PO, x}$	transportation time without order to the workstation picking order x
$ER_{PO, y}$	transportation time from the workstation picking order y to the goods receipt
ER_{ratio}	ratio of empty runs to runtime
ER_{total}	total empty runs of the means of transport
$ER_{WP, m}$	total transportation time to/from the workstation wedding/packaging 1/2 of means of transport m without transport order
$ER_{WP, x}$	transportation time without order to the workstation wedding/packaging x
$ER_{WP, y}$	transportation time from the workstation wedding/packaging y to the goods receipt
$ER_{WS, m}$	total transportation time to/from the workstation WS 1.1/1.2 or WS 2.1/2.2 of means of transport m without transport order
$ER_{WS, x}$	transportation time without order to the workstation x
$ER_{WS, y}$	transportation time from the workstation y to the goods receipt
f	number of defective or missing parts
mot_m	means of transport m
n	number of completed customer orders
O_i	customer order i
$O_{i,j}$	transport order j of customer order i
OS	order sequence
OS_c	order sequence of chromosome c

$OS_{c,s}$	solution value of the OS matrix of a chromosome
$p_{c,s}$	probability of chromosome c to be selected based on its solution value s
$p_{i,j}$	position of transport order j of customer order i in the global order
q	number of transport tours
r	random number
RT_{total}	total runtime of the means of transport
s	number of failures
S	solution set of all chromosomes in the population
s_c	solution value of chromosome c
SD_c	schedule deviation of order c
SD_i	schedule deviation of order i
s_m	proposed solution value of the means of transport m
t	simulation time
$TE_{i,j}$	transport end of transport order j of customer order i
TF_{total}	total failure time of the means of transport
$t_{m,i,j}$	transportation time of the means of transport m for transport order j of the customer order i
TPT	total throughput time of the city scooter
TP_{total}	total processing time of the city scooter
$TP_{WS,\,xj}$	processing time at workstation WS x for model j
$TR_{MOT,\,m}$	repair time of the means of transport m
$TS_{i,j}$	transport start of transport order j of customer order i
TT_c	transport time of order c
TT_{GI}	total transportation time to the goods issue
$TT_{GI,\,x}$	transportation time from the workstation wedding/packaging x to the goods issue
TT_i	transport time of order i
TT_{PO}	total transportation time to the workstation picking order 1/2
$TT_{PO,\,x}$	transportation time from the goods receipt to the workstation picking order 1/2
TT_{total}	total transportation time of the city scooter
TT_{WP}	total transportation time to the workstation wedding packaging 1/2
$TT_{WP,\,xy}$	transportation time from the workstation WS x to the workstation wedding/packaging y
TT_{WS}	total transportation time to the workstation WS 1.1/1.2 or WS 2.1/2.2
$TT_{WS,\,xy}$	transportation time from the workstation picking order x to the workstation WS y

TW_{DP}	total waiting time at the workstation WS 1.1/1.2 or WS 2.1/2.2 and wedding/packaging 1/2
TW_{GR}	total waiting time at goods receipt
TW_{RP}	total waiting time at the workstation WS 1.1/1.2 or WS 2.1/2.2 and wedding/packaging 1/2
TW_{total}	total waiting time of the city scooter
u	time of the observed period
UT_{GI}	total utilisation time to the goods issue
$UT_{GI, x}$	transportation time from the workstation wedding/packaging x to the goods issue
UT_{KR}	utilisation time of the KollRo
UT_{m}	utilisation time of the means of transport m
UT_{MOT}	total utilisation time of the means of transport
UT_{PO}	total utilisation time to the workstation picking order 1/2
$UT_{PO, x}$	transportation time from the goods receipt to the workstation picking order x
UT_{total}	utilisation time of the means of transport
UT_{WP}	total utilisation time to the workstation wedding/packaging 1/2
$UT_{WP, xy}$	transportation time from the workstation WS x to the workstation wedding/packaging y
UT_{WS}	total utilisation time to the workstation WS 1.1/1.2 or WS 2.1/2.2
$UT_{WS, xy}$	transportation time from the workstation picking order x to the workstation WS y
w	number of workstations

Chapter 1
Introduction

In this chapter, the reader is introduced to the research that was conducted. The chapter commences with a brief theoretical background. The reader gains a comprehensive understanding of the context of the research problem. Furthermore, it leads to the formulation of the research questions and the research objectives. Thereafter the scope of the research study is demarcated along with the research design and methodology. Finally, the chapter concludes with the outline of the research study.

1.1 Background and rationale of the research

Globalisation has changed global markets and environments significantly over the last years (Leitão, 2009, pp. 979–980). This change affects today's markets, leading to an ever-increasing complexity of logistical systems, and thus causing new demands on their design and control. Typical examples of these market-oriented trends are the increasing customer focus. This manifests itself in a change from the seller to the buyer market, the increasing networking of companies within the logistics chain and shorter product life cycles with simultaneously increasing variety (Scholz-Reiter, Windt, & Freitag, 2004, pp. 357–362).

Taking the automotive industry as an example, Dieter Zetsche, former CEO of Daimler AG, announced in 2014 that the company would consequently continue with its model initiative, hence further increase the variations of the offered models in all business areas over the next years (Daimler AG, 2014, p. 65). According to a press release of Progenium, Daimler AG is not the exception. In fact, the result of a study in 2014 by Progenium verified that the configuration options per vehicle model increased in average by 100 options from the 1990s to 2014 (Mandat, 2015, p. 2).

Further examples of the increase in the variant development of the automobile industry are presented based on Schlott (2005, pp. 38–39).

- At the Mercedes plant in Rastatt, exactly two out of 1.1 million A-Class vehicles were identical.
- At Ford in Cologne, 49 switches for instrument panels, 14 horns, 308 exterior mirrors, 92 mufflers and 13 fuel caps were installed.
- At Audi, the changes due to the conversion of the A6 model are shown in Figure 1.1.

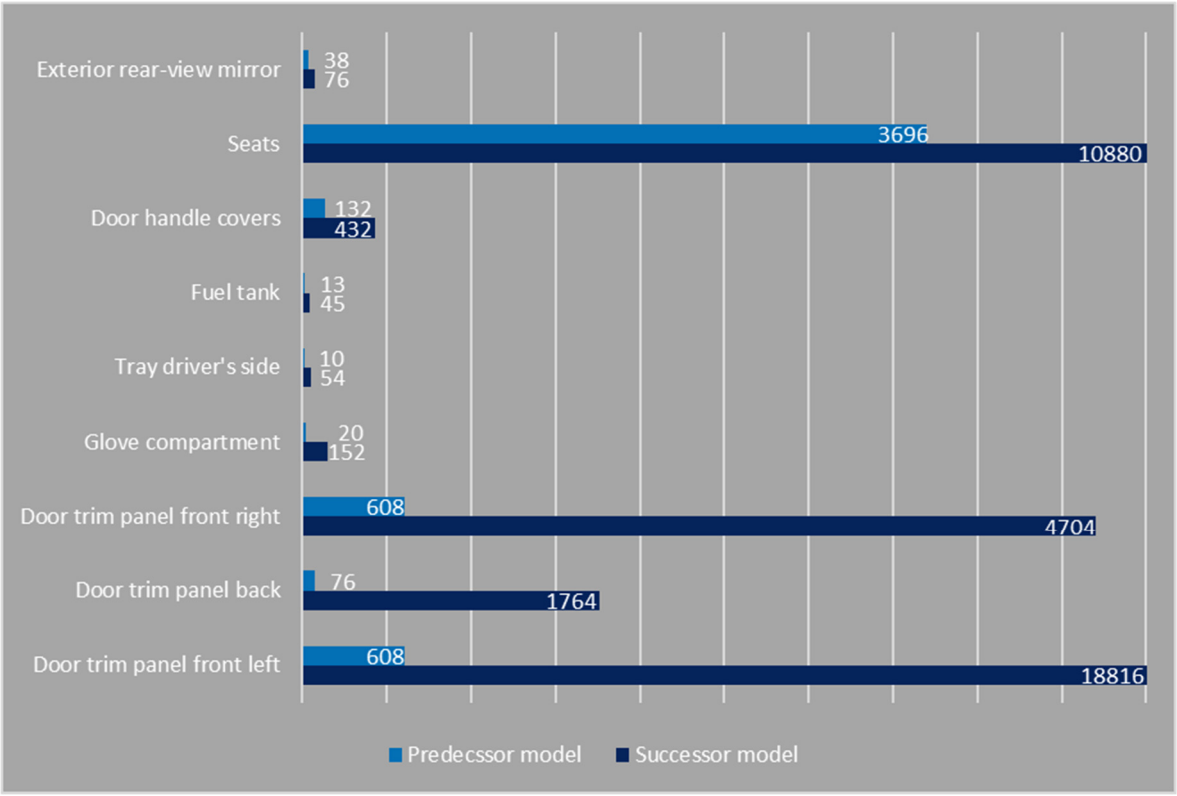


Figure 1.1: Example of the variant explosion in a vehicle model change (Schlott, 2005, pp. 38–39)

In order to tackle these challenges, the flexible concepts used in production planning and control are often based on the requirement to be able to adjust order sequences or machine allocation as quickly as possible (Niehues, 2016a, p. 69).

Consequently, these increased demands for flexibility in production systems are also relevant for the material supply, since a rescheduling of production processes is only possible if the availability of the required materials can be ensured. If the production facilities and processes can react to quantities that cannot be accurately predicted and to the equally unpredictable distribution of variant configurations, logistics is also directly affected by these fluctuations. A good scalability of logistics is therefore essential for the long-term economic operation of

production. The potential of the new concepts can only be fully exploited through the smooth interaction of all partial aspects (Teschemacher, 2019, pp. 1–2).

Coping with these new challenges requires logistics concepts and methods that ensure a high degree of flexibility and adaptability of the logistics system (Freitag, Herzog, & Scholz-Reiter, 2004, p. 23). Conventional logistics planning and control systems do not meet these requirements. Due to the increasing complexity of logistical systems, it is often not possible to provide all decision-relevant information to a central system in the required time and to derive control measures according to a defined target system. The central, sequential planning is based on strongly simplified premises (predictable lead times, static lot sizes, etc.), which only map reality insufficiently and have several weaknesses. The central, sequencing planning of conventional logistics systems takes place before the implementation and, in contrast to real-time planning, is based on an outdated and thus inadequate database. The system is not able to send feedback between the subsystems as well as change the created plan at runtime (Scholz-Reiter et al., 2005, pp. 1–2). Central logistics systems are too slow in terms of dealing with the planning and the control of order processing in their response to disturbances. These fundamental weaknesses of conventional logistics planning and control systems require a fundamental redesign of logistics systems.

1.2 Research problem statement and questions

The internal logistics of any production system have a significant impact on the material flow costs and thus the cost of a product (Fischer & Dittrich, 2004, p. 1). Aggteleky (1990, p. 491) estimates the material flow costs to be around 20–40 % of the cost of a product. Thus, logistics is one of the biggest cost drivers in production. This statement is supported by the fact that often the primary goal of layout planning in factory planning projects, is to minimise the transport volume (Scholz, 2010, p. 4). Hence, the internal material flow systems directly influence the profitability and competitiveness of a company (Fischer & Dittrich, 2004, p. 1). Nowadays, increasing product variety is automatically linked to an increase in cost and process time (ElMaraghy et al., 2013, p. 640).

The two target criteria, performance and costs, are relevant for internal logistics, which in many cases compete with each other. Performance is the ability to quickly adapt to changes in the logistics system and its processes as well as the short-term reaction to incoming orders and their delivery (Lieberoth-Leden, Röschinger, Lechner, & Günthner, 2017, pp. 483–484). On the other hand, it is still necessary to work within the cost optimum - an improvement of flexibility at the expense of economic efficiency is generally only acceptable to a limited extent. The necessity for efficient processes is shown in the logistics target triangle, which illustrates the prevailing conflict in logistics between the simultaneous desire for fast deliveries, high quality and thus few production disruptions caused by logistics and low costs (Arndt, 2015, p. 81). An improvement in two objectives can almost always be achieved by deterioration of the third (e.g. improvement of quality and delivery time by significantly increased costs), but an improvement of all objectives can usually only be achieved by more efficient processes.

In order to be able to react flexibly to the dynamics in production and logistics systems, any adjustments of the logistics system must not generate excessive logistics costs, as otherwise the profitability of such adjustments will be negatively influenced by the subsequent costs. The logistics system must be designed in such a way that the automated sequencing plan autonomously adapts to the dynamics in the production environment without further manual intervention (Teschemacher, 2019, pp. 4–5). Considering the described changes due to globalisation and the complexity of the problem, logistics planning and control systems have to consider different aspects. Therefore, in order to solve the above-mentioned aspects, the following primary research question (PRQ) and the underlying secondary research questions (SRQ) in terms of logistics planning and control will be answered in the course of this work:

PRQ:	**How can the control of intralogistics means of transport in production with individualised products be optimised in order to ensure a high target achievement?**

The results of the primary research question involve the development of a system to control the internal material flow that is able to automatically adapt to a dynamic production and thus respond robustly to disturbances. At the same time, due to competitive pressure, a high degree of logistical target achievement is essential. The aim of logistics planning and control is thus the exploitation of the performance potential and cost optimum of the company.

SRQ1:	How to manage intralogistics means of transport efficiently in a dynamic production?

The dynamic in logistics planning and control systems increases due to the amount of available information as well as the number of changes in the production schedule and disturbances. Therefore, robust behaviour of the system to counteract the dynamics is important for order processing to achieve a high degree of logistical target criteria (Freitag et al., 2004, p. 23; Rekersbrink, 2012, pp. 2–3; Scholz-Reiter, Freitag et al., 2005, p. 2). Also, the increasing structural complexity of the logistics processes combined with the dynamic aspect makes it more difficult to ensure that the relevant information can be retrieved at all times and at any location (Rekersbrink, 2012, pp. 2–3; Windt, 2008, pp. 366–367).

SRQ2:	How should the order processing be managed in order to achieve a robust system?

Another important aspect of control methods for sequencing planning is their computing time. If disruptions occur in production, it is necessary to reschedule the order processing as quickly as possible or to take the current situation into account for further decision-making (Aufenanger, 2009, p. 20). The faster the system is capable of setting up an updated sequencing planning of the orders, the more robustly the overall system behaves.

SRQ3:	How should the control of intralogistics means of transport be performed in order to achieve a high level of target achievement?

During the search for an optimal solution for the problem of order allocation to the various means of transport, the logistical targets must be considered equally. The quality of the found solution has a direct impact on, for example, logistics costs or logistics performance. Therefore, it is important to re-evaluate the order allocation depending on the situation in order to ensure a high degree of logistical target achievement. In practice, it is important to identify a good compromise between computation time, solution quality and robustness (Heger, 2014, p. 4).

1.3 Research objectives

By answering the above-mentioned research questions, this work aims to develop an autonomous system that provides an improved intralogistics sequencing control system which meets the new requirements and provides a high level of target achievement. Therefore, the research objective of developing an autonomous system is divided into four sub-objectives.

High efficiency: A high efficiency can only be achieved by using the right means of transport, at the right time and for the right transport order.

Holistic view: The described system must be designed in order to be able to consider the intralogistics system as a whole. Therefore, several different means of transport with their specific technical requirements and abilities will be included in the system. The focus of this thesis does not rely not on a single means of transport.

Flexibility of the overall system: Changes to the production facilities or processes must not lead to costly adjustments for the logistics concept. The system can react to external influences and thus optimally control its existing resources in order to reach a high target achievement. Since disturbances are to be expected in practical use, the system must be able to deal with them and offer corresponding measures for correction.

Real-time capability of the route calculation: The routes must be calculated as quickly as possible in case of new orders or changes of the overall system, e.g. disturbances like a failure of a means of transport so that a new, optimised sequence for all means of transport can be generated without any considerable delay.

In summary, the fulfilment of the above-mentioned objectives enables the system to react dynamically to changes by intelligent, short-term control of the means of transport. The developed system must be able to adapt to unpredictable changes without major effort.

1.4 Research design

In logistics research, there are mainly two factors influencing scientific work; economy and behaviour. Thereby, the goal of the economic approach is to minimise cost and maximise profit by using methods like cost analysis, mathematical modelling, simulation, and sensitivity analysis. The behavioural approach, in its turn, deals with the psychological and sociological aspects of situations through questionnaires, interviews, and case studies (Mentzer & Kahn, 1995, p. 232).

Economical and behavioural orientation have their foundation in the philosophy of positivism regarding scientific research. According to Mentzer & Kahn (1995, p. 232) positivism is defined as follows:

> *"Positivism has the goal to explain and predict reality, where reality is considered to be deterministic and reactive. Research findings in the positivist tradition are considered value-free, time-free, and context independent, with the general agreement that causal relationships can be discovered. Positivist researchers consider themselves separate from the research setting and at a privileged point of observation."*

This research study adopted the positivism research philosophy. The validity of the developed system is based on observations and evaluations carried out in the course of this work. The system was, therefore verified and validated in a simulation study.

In logistics research, deductive positivism is often discussed as the predominant research approach (Mentzer & Kahn, 1995, p. 232). The logic of the deductive model is aimed at a specific case, derived from a general law (Danermark, 2002, p. 84). This means that the deductive approach, according to Hyde (2000, p. 83), is a theory-testing process, which commences with an established theory or generalisation and then seeks to test whether the theory applies to specific instances. The researcher empirically tests the theory and confirms or rejects the general conclusions (Stentoft Arlbjørn & Halldorsson, 2002, p. 24). The goal of deductive research is to develop a theory, then to test it, to generalise it and thus to be closer to the true statement about reality. The generalization, built on previous knowledge, may result in the acquisition of the new knowledge (Peter & Olson, 1983, p. 120). The aim of deduction is to test a theory derived from these theories, which has been developed from previous empirical research (Danermark, 2002, p. 82).

This research study followed a deductive approach, testing a theory derived from the research problem. Thereby, the terms validity, reliability, and precision, according to Mentzer & Kahn (1995, pp. 238–240), are taken into consideration in this thesis to ensure the acceptability of the study findings by both the individual researcher and the research community.

The research design is divided into two parts, a literature review of the state-of-the-art research and simulation to provide high validity of the research. The literature review provides both historical perspective of the respective research area and present previous research endeavours. As a starting point for the deductive and positivist research design, the literature review is established by scientific research and then the formulated theory is tested.

Thereafter, a simulation study is conducted which is able to prove the theory by manipulating, controlling and measuring the input and output variables. Also, the simulation considers the size of the change and the relative importance of its findings. This means that different scenarios will be displayed in the simulation and the effects on the variables and how they will occur, are observed. To assess the practical feasibility of the system and to make recommendations to the research, the system is finally implemented in the Werk150, the factory of the ESB Business School on the Reutlingen University campus.

1.4.1 Literature review

The literature review is the starting point of the research, which justifies why review papers are frequently cited. It generates ideas of research and summarises existing research by identifying patterns, themes and issues (Easterby-Smith, Thorpe, & Lowe, 2002, pp. 159–160). Also, the study includes a review of previous research, whether from conceptual or empirical work, and has to be checked against existing theories (Saunders, Lewis, & Thornhill, 2009, p. 46).

1.4.1.1 Process of the literature review

The literature review is based on a defined systematic approach. It is based on Mayring's statement that the research is driven by approaching the material in advance, determining the conditions and following a clear process, as this allows conclusions to be drawn on the analysed material (Mayring, 2015, p. 50).

The thesis uses a modified process model for the literature review according to Kotzab & Westhaus (2005, p. 94), which comprises the following steps:

- Material collection: The material to be collected is defined and demarcated. This might include taking a look at how the material was obtained.
- Category selection: Structural dimensions and related analytic categories are selected, which are to be applied in the literature review to structure the field. Structural dimensions form the major topics of analysis, which cover various analytic categories.
- Material evaluation: The material is analysed and sorted according to the structural dimensions and categories built. This should allow identification of relevant issues and interpretation of results.

1.4.1.2 Delimitation and limitations

For a literature review it is particularly important to define clear boundaries to demarcate the research. In this context three provisos are made:

- The work can be categorised according to literature in the field of intralogistics control. This means that the thesis focuses on the planning, control and monitoring of the material flow from the goods receipt throughout the entire production process up to goods issue, e.g. warehouse for finished goods. Processes before the goods receipt and after the goods issue are not considered.
- The focus is on transport logistics and thus on the section's transport organisation and means of transport. Publications dealing mainly with cargo handling, packaging logistics, warehouse logistics and information logistics are not considered.
- Articles focusing only on ethical or environmental demands placed on intralogistics topics are excluded.

This analysis aims at scientific publications with clear conceptual or empirical content. Practitioner papers, which only provide anecdotal evidence were not to be considered. The relevant period was set from the year 2016 onwards.

The work presented is based on literature research, where German and English publications were analysed, including books, edited volumes, and journal papers.

First of all, the main source for the literature research are Reutlingen Library, Datenbank-Infosystem Hochschule Reutlingen, elektronische Zeitschriften Bibliothek, Stellenbosch Library, and Google Scholar.

Selected journals, published in English or German, were seen as particularly relevant; e.g. International Journal of Logistics Management, International Journal of Physical Distribution & Logistics Management, Journal of Business Logistics, Logistics Journal and Logistik entdecken. Furthermore, three databases were used to search for further articles, such as those provided by major publishers; e.g. Elsevier (www.sciencedirect.com), Emerald (www.emeraldinsight.com), Fraunhofer-eprints (www.eprints.fraunhofer.de). In this way, related edited volumes and single papers in other journals were also found. As an additional step, literature cited in identified papers was checked.

1.4.2 Simulation

Simulation can be compared with formal modelling and empirical observation and experimentation (Axelrod, 1997, p. 17). Conway, Johnson, & Maxwell (1959, pp. 106–107) understood simulations as statistical experiments. According to the authors, the simulation and experiment methods share the characteristic of being able to manipulate and control variables within the simulation.

Simulations allow experimentation on a computer-based model to investigate cause-effect relationships which cannot be observed in reality because of cost or time restrictions, safety issues, or legal requirements. The precise replication of a situation is rarely possible with an experiment, therefore the use of simulation is helpful (Pidd, 2004, pp. 7–8). Due to the inherent complexity of reality, empirical studies or experiments cannot successfully and completely control the affected parameters. Hence, idealisation in form of simulation is sometimes seen as the primary way towards scientific progress (McKelvey, 1999, p. 5).

The theory derived from the literature review is later tested and confirmed during a simulation study and afterwards in a field-based experiment in the Werk150, the factory of the ESB Business School on the Reutlingen University campus. The simulation should be able to prove the theory by manipulating, controlling and measuring the variables. During this process sufficient numerical size of quantitative data is collected in order to generalise a conclusion statistically. Within the experiment, the system is implemented in a running production system, namely the Werk150, to prove its practical use.

1.4.2.1 Process of the simulation

The simulation unites the aspects of mathematical modelling and empirical research. This achieves high generalisation, practical relevance and internal and external validity (Kotzab & Westhaus, 2005, p. 447). In the following section the process model for the implementation of a simulation based on Böse (2012, p. 102) is described. The present work is structured according to this process shown in Figure 1.2.

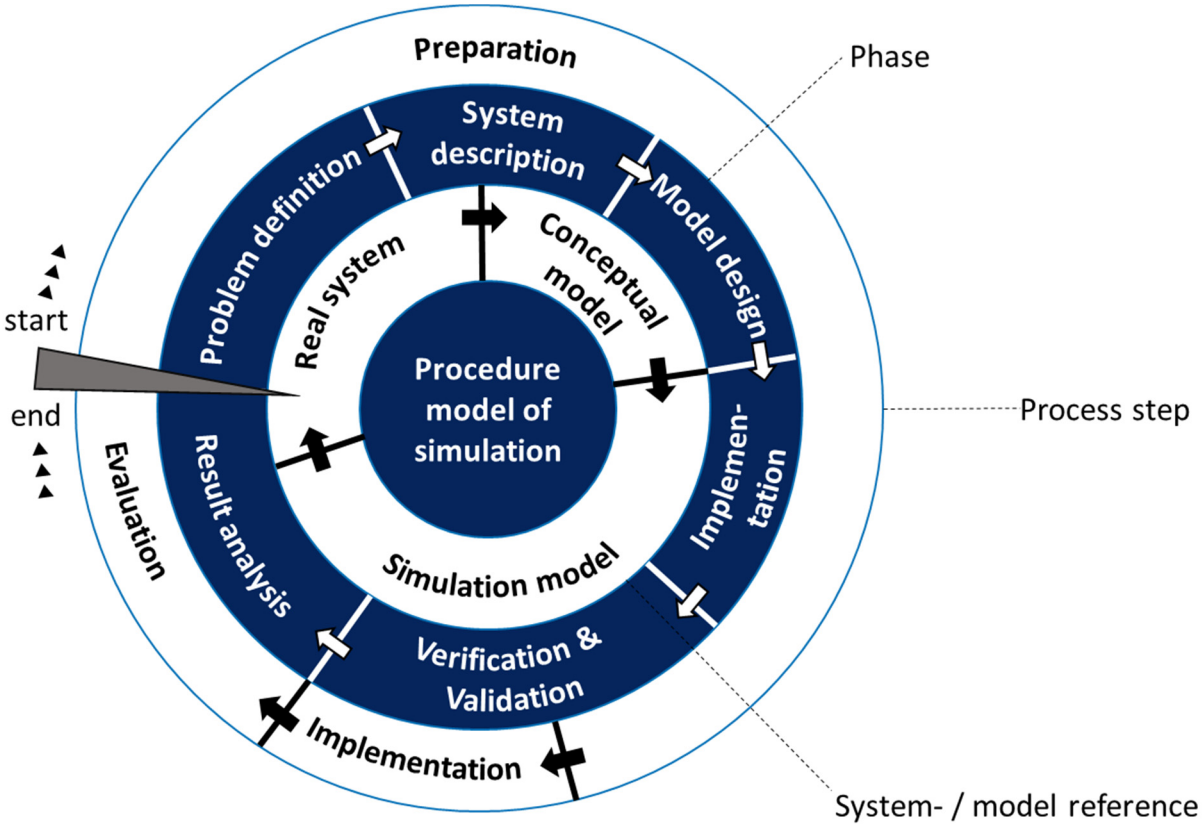


Figure 1.2: Process model of simulation based on Böse (2012, p. 102)

In general, the simulation includes the preparation, implementation, and evaluation of a targeted experiment with a simulation model. These phases of the simulation, represented in the outer ring of the procedure model, contain the individual working steps of the simulation, which are depicted in the middle ring of the procedure model. The inner ring of the procedure model finally illustrates the system or model reference.

Following the formulation of the problem and the objective derived from the simulation study, the considered system with its subsystems and boundaries are defined. In the framework of the model design, the processes of the real system are transferred into a conceptual, not yet experimental, model. With the help of simulation software, this conceptual model is implemented as a simulation model. In the course of the verification and validation (V&V)

process, the accuracy and plausibility of the mapping of the real system in the simulation model is checked. Next, during the second phase of the simulation, the simulation experiment is carried out. This is followed by the evaluation phase, which includes the analysis and evaluation of the simulation results, and the transfer of the results to the real system.

Problem definition: At the beginning of a simulation study, simulation problems and the objectives to be achieved are described. Based on the description of the simulation problems and objectives, a target system needs to be defined, which contains parameters that characterise the behaviour of a system with respect to its task. A target system usually consists of an overall goal and several, partly complementary, partly competing, sub-goals. For example, to assess control strategies of a transport logistics system, the global goal of maximising profitability can be broken down into the sub-goals of 'high adherence to schedules', 'short order waiting time and transportation time', 'high utilisation and transportation frequency' as well as 'minimum empty runs and high flexibility'. According to VDI 3633, the exact formulation of problems and objectives is essential for the further course of the simulation study as it forms the basis for the demarcation of the system within the framework of the model design, the performance of the simulation experiments as well as the interpretation of the simulation results (Verein Deutscher Ingenieure, 2018, p. 12).

System description: The basis for the development of the conceptual model is a description of the considered technical system. For this purpose, a system demarcation is carried out; system boundaries to the system environment are drawn up and the system input and output variables are defined. The system demarcation is set by isolating the view of the considered process from the global process chain. Only the system elements, structures, processes, as well as input and output variables which are relevant to the research are described (Gunther, 2000, p. 16). The system demarcation thus depends on the problem and ignores less important aspects of the real system (Page, 1991, p. 2).

Model design: The model design refers to the transformation of a real or imaginary system into a conceptual, not yet experimental model. In the context of simulation, the standard VDI 3633 defines the term 'model' as a simplified replication of an existing or imaginary system with its processes in a conceptual or representational system (Verein Deutscher Ingenieure, 2018, p. 19).

Determining the level of detail of the model is of central importance in modelling since this significantly determines the degree of conformance of the simulation results with the actual

behaviour of the real system (Mehl, 1994, p. 1). In the case of a low level of detail, the explanatory power of the model decreases or results in wrong findings. On the other hand, a high level of detail results in an increase in the modelling effort, runtime, error rate and interpretation effort of the results (Gunther, 2000, p. 22). According to the standard VDI 3633, in summary, the replication of the system in the model should be as accurate as necessary for the given research objectives, but as abstract as possible (Verein Deutscher Ingenieure, 2018, p. 21).

Implementation: The developed conceptual model is implemented into an experimental simulation model. With the help of a simulation software or simulation language, the dynamic behaviour of the considered system and its processes is replicable and executable.

Verification & Validation: Before a simulation model is used for simulation experiments, it is important to evaluate the validity of the model. The verification and validation techniques examine whether the created simulation model accurately reflects the conceptual model and shows an error-free behaviour of the real system with regard to the analysed objectives and sufficiently detailed replication (Page, 1991, p. 16).

Result analysis: In the context of the result analysis, it is necessary to prepare, interpret and evaluate the results of the simulation runs. The evaluation of the simulation results is based on the original question and objective of the simulation study. Limitations of the model must be taken into account regarding the interpretation of the results (Page, 1991, p. 17).

1.4.2.2 Delimitation and limitations

In the following discussion, assumptions will be made in order to establish the framework conditions of the developed system and thus enable a targeted approach. Some assumptions are made with regard to the system environment:

Control system: In this thesis, the control of intralogistics means of transport is considered only in the area of production logistics. The area of application of the system to be developed covers exclusively the production-accompanying period from goods receipt to goods issue. Furthermore, it is assumed that the necessary resources are available for all released customer and transport orders.

Production programme: The production programme is unknown for the period under review. However, the data required for an order is always complete and up to date. It can be assumed that the planned values for machining and setup times are realistic. The processing times of the orders are fixed and usually cannot be accelerated.

Directional machine sequence: The sequence of the processing stations for an order has a directional sequence. It is not possible for the product to be transported to the workstation 2.2 (WS) before it has been to WS 1.2, for example. Each order has a fixed source-sink relationship. An order passes through each workstation only once. Return flows do not exist and are therefore not considered in the design of the control system.

Limited machine capacity, unlimited personnel and storage capacity: Each machine can process only one order at a time. In contrast, restrictions regarding personnel and storage space are not taken into account, as it can be assumed on the basis of production planning that every machine is operated and that the number of orders in production is already limited in the planning phase to a degree that the available space is sufficient for storage.

Specifications of the means of transport: For purposes of simplification, routes and means of transport are not taken into consideration with regard to their dimension length, width and height. This means that theoretically, any means of transport can reach the desired pick-up location and delivery stations. Excluded from this rule are only stationary means of transport, e.g. conveyor belts, which are assigned to determined pick-up and delivery stations. In addition, the means of transport operate in a collision-free environment. The loading and unloading of the means of transport is not considered in detail in the simulation. For the loading and unloading processes only reference values are used, which represent reality as closely as possible. Focusing on the human as a logistical means of transport, characteristics like motivation, selfishness-altruism or stress are not taken into account. Considering logistical vehicles, charging processes for the battery are not included in the simulation.

1.5 Research study outline

Following the introduction in this chapter, the thesis is embedded in its thematic background in Chapter 2. Therefore, the observed organisational structures in this paper are first defined. This is followed by an explication of the term autonomous control in logistics and the presentation of a catalogue of criteria for describing autonomous control in logistics systems. In addition, the problem of sequence planning in intralogistics, which is also known as vehicle routing problem (VRP) is presented. To solve the VRP, optimising methods and heuristic methods are commonly used in scientific studies. The methods are based on an intralogistics target system, which is described in the following section. Furthermore, the Werk150, the factory of the ESB Business School on the Reutlingen University campus, is presented as an application for flexible flow production. The developed autonomous system will later be implemented and validated in the Werk150. Therefore, the production system of the Werk150 serves as a model for this thesis.

In Chapter 3, criteria for the structured content analysis are defined. Then, recently published scientific work is summarised and evaluated according to these criteria. The chapter finally presents the main findings of the literature review and the derived research gap.

In Chapter 4, the problem definition of the simulation study is first specified, which determines the questions to be investigated and the objectives to be achieved. A target system is then defined for the considered application scenario.

Chapter 5 commences with the theoretical basics of system theory. For the simulation study, a system model is generated. The abstract model describes the important subsystems as well as the system elements for the development of the system. With the help of the system model, the scope of the simulation study is defined.

Next, the basics for modelling are described in Chapter 6. Based on the system model developed in the previous chapter, the class diagram for the system is then created. The class diagram determines the characteristics and relation of the objects in the system. Furthermore, an algorithm is developed to optimally allocate customer orders in production to the available means of transport. According to the logistics target criteria mentioned in Chapter 3, the algorithm considers several objective functions. The applied approach is based on a genetic algorithm with neighbourhood search to iteratively search the solution space.

Chapter 7 describes the implementation of the algorithm into the simulation software. First, the basics of simulation are explained and from there the simulation method used is derived.

Subsequently, the developed simulation structure and the control of the individual agents within the system are explained. In addition, the interaction and communication between the agents is described.

This is followed by the V&V process which is described in Chapter 8. First, different theoretical V&V techniques used in this research study are described. Then, the implementation of these techniques in the simulation study is explained. Furthermore, the practical validation of the presented system takes place. The aim of this work is to prove the practical feasibility by a detailed investigation of the transfer into an existing production, the Werk150, the factory of the ESB Business School on the Reutlingen University campus. For this purpose, the hardware used and the software components of the Werk150 are described. After the implementation of the developed system, the practical validation takes place in the Werk150.

In Chapter 9, the system behaviour of the static system and the autonomous system is analysed within the framework of the result analysis. The obtained data from the simulation results are then evaluated in terms of the defined objectives of the simulation study. The relevant logistical target criteria are calculated, compared and evaluated.

Chapter 10 contains a summary and a conclusion of this research study and briefly discusses the key findings. In addition, an outlook is provided into further potential research projects with respect to the research questions considered in this thesis.

A graphic illustration of the structure of this thesis is shown in Figure 1.3.

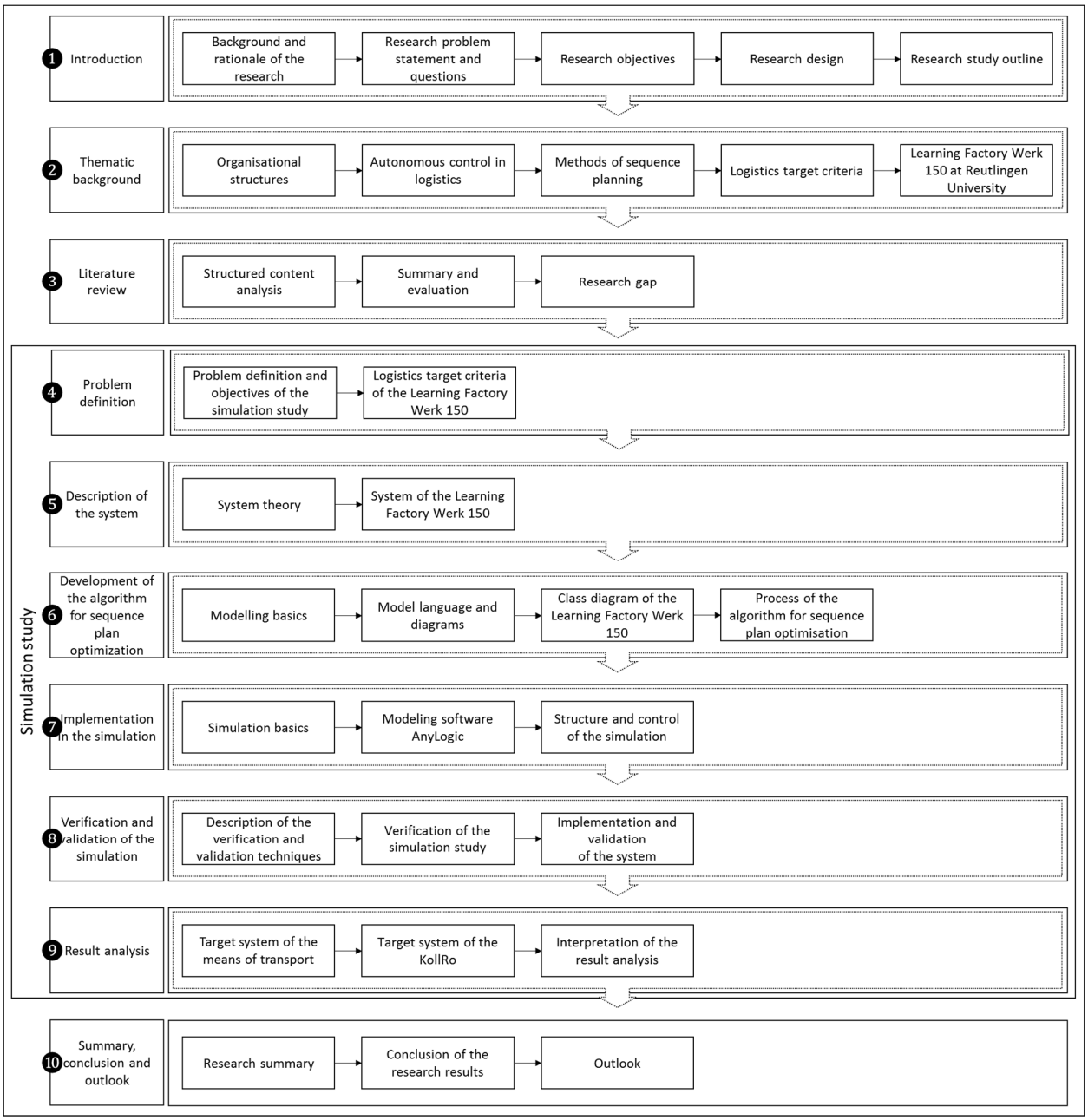


Figure 1.3: Outline of the research study

Chapter 2
Thematic background

In this chapter, the organisational structures considered in this paper are presented and their complexity described. In addition, the theoretical basics for autonomous control of logistics systems are explained. A catalogue of criteria for the description of the characteristics of autonomous control as well as for the determination of the degree of autonomous control in logistics systems is presented. Furthermore, existing methods for sequence planning in intralogistics are examined. These approaches are subdivided into optimising methods and heuristical methods. In addition, a target system for intralogistics is described, which forms the basis for the further course of the work. Furthermore, the Werk150, the factory of the ESB Business School on the Reutlingen University campus, is introduced as a flexible flow production and serves as a model for the developed system.

2.1 Organisational structures

Various organisational structures have evolved in manufacturing in the course of industrial development. The increasing variety and number of products and the different influences have an impact on the structure of an organisation. The various influencing factors within a company can be subdivided into four groups (Wiendahl, 2010, pp. 27–29):

- The organisational structure of a company depends on the orientation of the company towards the market or the customer. The aim of a company is to ensure the achievement of the target criteria delivery reliability and delivery time, as well as costs, quality and flexibility. In the case of changing market conditions, the company should be able to react as quickly as possible on the new demands.
- The personnel structure of a company is another influencing factor in terms of the qualifications of employees and the technologies used. Furthermore, the production of a company must be economical and flexible.
- Technological influences are related to the design of the products including geometric shapes, dimensions and tolerances as well as the material used.
- The economic factors can be derived from the market orientation and the technology used, with the lot size as the most important influencing factor. The lot size has a major

impact on economic efficiency as the utilisation and setup times of the machines depend on it, as well as the costs of warehousing and tied capital. Another influence on the structure of an organisation is the repetitive character of production. Single or multi-model production lead to different demands on the production system.

The problem of sequence planning depends strongly on the organisational form of the company, since different organisational forms have different degrees of decision-making and freedom. This affects the requirements and complexity of sequencing. In the following section, the organisational forms considered in this thesis are examined in more detail.

- *Workshop production:* In a workshop production, the required resources are stationary and the workstations are arranged according to the processing methods into organisational units (workshops) (Dangelmaier, 1999, p. 315). Workshop production enables flexible adaptation to different products and their different processing sequences (Wiendahl, 2010, p. 33). The disadvantage of workshop production is that the products must be transported to their technologically-related workshops. As a result, it is usually not possible to precisely match the work and transport processes of the individual orders and thus avoid waiting times for processing or onward transport (Günther & Tempelmeier, 2003, pp. 14–15).
- *Flow production:* In a flow production, the products are grouped together into product-related units. The different stations of a production line represent the individual workplaces. The sequence of processing steps of individual organisational units is product-specific. Within the organisational units, there is a fixed sequence with generally unified cycle time (Dangelmaier, 1999, p. 317). Buffers are typically installed between workstations to absorb minor disturbances at individual stations in order to maintain a robust system. Furthermore, a close coordination of the capacities of the system is required. Disadvantages of a flow production occur for example in the case of technical changes. Workstations can be re-designed only with great effort, since the workstations are installed for a specific product. Another disadvantage of flow production arises in cases where the desired product cannot meet the predicted demand, hence the utilisation of the facilities cannot be used economically (Wiendahl, 2010, pp. 31–32).
- *Flexible flow production:* With flexible flow productions, the aim is to combine the efficiency and transparency of static flow productions with the responsiveness and

> flexibility of workshop productions to achieve a high degree of fulfilment of the target criteria. In this case, finding alternative ways of production in case of machine downtime or dealing with defective parts is a major challenge (Heger, 2014, p. 17). Flexible flow production is a general concept which relies on automated product and information flows, enabling automatic, non-clocked, direction-free and thus highly flexible production of a defined group of similar parts (Wiendahl, 2010, p. 36).

For the flexible flow production as an organisational structure in production, logistics planning and control systems often do not meet the required flexibility and efficiency of order processing. The main reasons are the high complexity and dynamics of business processes that characterise flexible flow production. To cope with the complexity and dynamics in flexible flow production, the use of autonomous control represents a new approach for control in intralogistics. The theoretical basics of autonomous control in logistics are explained in the next section.

2.2 Autonomous control in logistics

In contrast to central planning and control systems, autonomous control systems distribute the decision-making process to the individual logistics objects. Logistics objects include both physical objects such as means of transport or products and intangible objects like customer orders or transport orders of a logistics system. The structure of an autonomous control system has a decentralised, heterarchical design compared to centralised, hierarchical system structures with conventional external control.

The term autonomous control can be defined according to Windt & Hülsmann (2007, p. 8) as follows:

> *"Autonomous Control describes processes of decentralized decision-making in heterarchical structures. It presumes interacting elements in non-deterministic systems, which possess the capability and possibility to render decisions. The objective of Autonomous Control is the achievement of increased robustness and positive emergence of the total system due to distributed and flexible coping with dynamics and complexity."*

The definition for autonomous control describes the maximum degree of autonomous control in logistics systems. However, a maximum degree of autonomous control would result in chaotic system behaviour. Consequently, the logistics system will contain aspects of externally controlled and internally controlled objects in order to ensure a high target achievement.

2.2.1 Characteristics of autonomous control

According to Böse (2012, pp. 14–15), the autonomous control of logistics systems can be subdivided into the following characteristics: decentralised decision-making in heterarchical systems, interaction of the logistics objects and non-deterministic system behaviour and positive emergence. In the following paragraphs, the characteristics of autonomous control in logistics will be discussed in more detail.

Decentralised decision-making in heterarchical systems: The organisational structure of an autonomous control system is heterarchically structured. Thus, autonomous control enables the individual logistics objects to make decisions independently. The individual logistics objects are able to change the global target prioritisation based on their local target system. The target system of the logistics objects is dynamic, since the target prioritisation can be changed over time in the process flow.

Interaction of the logistics objects: The ability of interaction among the logistics objects requires the availability of information. This interaction can be at various levels depending on the degree of autonomous control. The interaction of the logistics objects based on the ability to access data provided by other logistics objects is characteristic of a low degree of autonomous control. Communication, in the form of the exchange of data between the logistics objects, requires a higher degree of autonomous control of the system. The highest form of autonomous control in terms of interaction ability is the ability of the autonomous logistics objects to coordinate activities.

Non-deterministic system behaviour and positive emergence: The system aims to increase its robustness and achieve a positive emergence through a distributed and flexible control of dynamics and complexity in logistical systems. The term non-determinism means that the system's behaviour cannot be predicted exactly; instead several possibilities exist for the transition into a subsequent state with the same input of initial variables. A higher robustness of the system can be achieved by a fast reaction to unpredictable, dynamic influences on the system. In the context of autonomous control, positive emergence means that the interaction of the individual logistics objects generates a better target achievement for the overall system. Thus, individual logistics objects such as means of transport, products, etc. ensure a higher target achievement of the overall system than a central target achievement approach.

2.2.2 Criterion catalogue for determining autonomous control

In this section, the criterion catalogue of Böse (2012, pp. 24–26) for determining the degree of autonomous control of a logistics system is presented. The criterion catalogue for determining the logistics system is represented in the form of a morphological box. The criterion catalogue includes three categories; decision-making, information processing and decision execution. The criterion catalogue allows a first analysis of the logistics system based on 17 criteria of autonomous control and the corresponding degree of fulfilment. The criteria and their degrees of fulfilment are generally formulated in order to classify logistics systems on different aggregation levels and logistics areas, e.g. production or transport logistics. Figure 2.1 shows the criteria and their degrees of fulfilment of autonomous control in the form of a morphological box, where the degree of fulfilment of the individual criteria is represented by the increasing degree of autonomous control from left to right. In order to assess the degree of autonomous control of logistical systems regarding the individual criteria, a value was assigned for each criterion depending on the fulfilment in the range of 0 (no autonomous control) to 3 (full autonomous control). By multiplying the appropriate value by the weighting factor for each criterion and adding up the values of all the criteria, the degree of autonomous control of the logistics system can be determined on a scale from 0 (no autonomous control) to 105 (full autonomous control).

Category C_i	Criterion C_{ij}	Weighting W_{ij}	Fulfilment of criterion F_{ij}			
Criteria of decision-making Initiation of the decision-making process, identification and evaluation of the decision alternatives, instruction and control of the selected decision alternatives	Maturity of the target system	2 (expected criteria)	long-term 0	medium-term 1.5		short-term 3
	Distribution of the target system	2 (expected criteria)	global 0	mainly global 1	mainly local 2	local 3
	Adaptivity of the target system	2 (expected criteria)	static 0	mainly static 1	mainly dynamic 2	dynamic 3
	Organisational structure	3 (must criteria)	hierarchical 0	mainly hierarchical 1	mainly heterarchical 2	heterarchical 3
	Number of decision alternatives	1 (preferable criteria)	none 0	some 1	many 2	unlimited 3
	Number of decision-making processes	2 (expected criteria)	few 0	some 1	many 2	a great many 3
	Type of decision making	2 (expected criteria)	extrinsic 0	intrinsic (rule based) 1.5		intrinsic (learning) 3
	Location of decision-making	3 (must criteria)	system level 0	subsystem level 1.5		System elements level 3
	Predictability of the system and element behaviour	2 (expected criteria)	elements and system deterministic 0	elements not-/ system deterministic 1	system not-/ elements deterministic 2	elements and system not deterministic 3
Criteria of information processing Collection, storage, transformation and transmission of information	Place of data storage	2 (expected criteria)	central 0	mainly central 1	mainly decentralized 2	decentralized 3
	Place of data processing	2 (expected criteria)	central 0	mainly central 1	mainly decentralized 2	decentralized 3
	Ability to interact	3 (must criteria)	none 0	data provision 1	communication 2	coordination 3
	Data transfer volume	2 (expected criteria)	low 0	intermediate 1	high 2	very high 3
Criteria of decision-execution Implementation of the decision made at the material flow level	Flexibility	1 (preferable criteria)	inflexible 0	slightly flexible 1	flexible 2	very flexible 3
	Identifiability	3 (must criteria)	no elements identifiable 0	few elements identifiable 1	many elements identifiable 2	all elements identifiable 3
	Measurability	2 (expected criteria)	no measurement 0	external measurement 1	internal measurement 2	external and internal measurement 3
	Localisability	1 (preferable criteria)	no elements locatable 0	few elements locatable 1	Many elements locatable 2	All elements locatable 3

Degree of autonomous control

$$\sum_{i=0}^{n}\sum_{j=0}^{m} W_{ij} * F_{ij}$$

C_i = Category
C_{ij} = Criteria
W_{ij} = Weighting
F_{ij} = Fulfilment of criterion

Figure 2.1: Criteria and specifications of autonomous control (Böse, 2012, p. 25)

The criterion catalogue presented above allows a qualitative statement about the degree of autonomous control of a logistics system. Based on the achieved score of a logistics system, a higher score indicates a higher degree of autonomous control. By using the criterion catalogue, different logistics systems can be compared in terms of their degree of autonomous control.

Various methods for the control of static and autonomous systems are described in the literature. In practice, a flexible flow production represents a non-deterministic polynomial-time hardness problem, called an NP-hard problem, in terms of the control of logistics systems regarding sequence planning. Consequently, optimal plans that have a realistic dimension cannot be created in a reasonable period of time, hence mathematical models have little relevance in practice. Heuristic methods, which try to reach a good achievement of the target criteria in a reasonable time, without guaranteeing an optimal solution are mainly used in practice (Heger, 2014, pp. 2–3). Therefore, various methods of sequence planning are introduced in the following section.

2.3 Methods of sequence planning

In order to be able to optimise the internal logistics processes, a description of the situation that can be interpreted by algorithms is required. The optimisation of the means of transport in internal logistics systems can be modelled as so-called vehicle routing problem (VRP) (Gyulai, Pfeiffer, Sobottka, & Váncza, 2013, p. 127). This is based on the travelling salesman problem (TSP), in which a traveller must select the sequence for visiting several cities in order to keep travelled distances to a minimum and finally return to the point of departure (Korte & Vygen, 2018, p. 623).

The VRP based on the TSP is a problem first described by Dantzig & Ramser (1959). It essentially defines the task of sequencing an existing quantity of orders e.g. customer orders or purchase orders and grouping them into individual tours to optimise a certain objective function, e.g. minimising the distance. Thereby, each tour is carried out by a single means of transport (Wenger, 2010, p. 40).

There are numerous existing approaches solving the VRP. A distinction is made between optimising solution methods, which are guaranteed to determine the optimum solution, and heuristic methods, which determine a good solution in a significantly shorter time but do not necessarily achieve the optimum. In the following section an overview of existing methods will be presented. Optimising methods calculate optimal solutions; thus these solutions meet the selected target criterion best. However, this only applies to deterministic problems where

complete information is available at the time of computation. Many optimising methods ignore the fact that most practical problems are characterised by dynamic influences (Heger, 2014, p. 33).

Heuristic methods usually do not guarantee an optimal solution but require less computation time and are therefore more frequently used in practice. They can be differentiated into constructive and iterative search methods. The constructive heuristics generate solutions without changing previously calculated partial solutions. Iterative search methods or metaheuristics like simulated annealing (SA), Tabu search (TS), genetic algorithms (GA), or bio-analogue methods, for example the ant colony optimisation (ACO), search for and gradually improve solutions according to certain principles (Heger, 2014, p. 34).

Furthermore, methods can be subdivided according to their degree of centralisation or decentralisation. The central processes, such as the greedy search algorithm, calculate an order schedule centrally for the production. This has the advantage that mutual dependencies can be better taken into account. However, a new overall plan must be created for each occurring change. In decentralised control systems, the calculation of the order schedule is distributed among the agents in the system. The decision-making is therefore decentralised, reducing the complexity of the sequencing problem. Thus, a better control of the complexity and dynamics to which the production systems are exposed is possible (Rekersbrink, 2012, p. 4; Rekersbrink, Scholz-Reiter, & Zabel, 2010, pp. 254–256). According to the author, the most important procedures and concepts regarding intralogistics are presented in detail below.

2.3.1 Optimising methods

The goal of production and logistics is to fulfil the target criteria optimally so that ultimately neither resources nor unnecessary time is wasted. Due to the high complexity of the various sequence problems, practice-relevant problems cannot be solved in a reasonable time. Optimising methods can nevertheless be used in various applications. Optimising methods are able to evaluate the solution quality of heuristics on the basis of small scenarios as well as calculating optimal partial solutions, which are then compiled into what is generally not an optimal overall solution (Heger, 2014, p. 35).

Branch-and-bound: The branch-and-bound method is based on the work by Land & Doig (1960, pp. 497–520) and in the field of production planning on Ignall & Schrage (1965, pp. 400–412). Recent work investigating sequence planning in flexible flow production has been carried out by Shchekutin, Short, & Overmeyer (2017).

The idea of the branch-and-bound method is to examine possible solutions using a decision tree but trying to avoid reviewing all possibilities, since individual solutions may already be excluded because of their partial solutions. For this purpose, a tree structure is dynamically generated, representing all possible solutions. Each node represents a partial sequence plan, not including all operations to be performed. The leaves of the tree represent a complete sequence of plans consisting of the partial plans of its predecessors. Splitting the total solution set into discrete optimisation sub-problems is called branching. Subsequently, the limiting of the possible solutions takes place, also referred to as bounding. Therefore, an upper bound is calculated, which is determined by the quality of the best solution so far. For the individual nodes of the decision tree, a lower bound is estimated. The order in which the nodes are selected is often determined by the depth search. By determining the lower bounds, the examination space is steadily reduced. Once the lower bound of a node of the decision tree exceeds the upper bound, the partial sequencing plan of this branch cannot be part of the optimal solution (Aufenanger, 2009, pp. 30–31; Błazewicz, Ecker, Pesch, Schmidt, & Weglarz, 2007, pp. 33–34; Brucker, 2007, pp. 56–60).

To solve the vehicle routing problem Fischetti, Toth, & Vigo (1994) and Desrochers, Desrosiers, & Solomon (1992) use a branch-and-bound algorithm taking into consideration different constraints. Laporte (1992) shows an overview of papers using the branch-and-bound algorithm to solve the vehicle routing problem.

Mixed-integer optimisation: In the linear optimisation, both the objective function and all restrictions of an optimisation model based on linear combinations of the decision variables are shown. Considering linear constraints expressed by equations or inequalities, it is important to minimise or maximise the objective function. In practice, many problems cannot be represented by continuous variables, for example, products cannot be split. Therefore, integer conditions are added if necessary for some of the variables in a linear optimisation model (Suhl & Mellouli, 2013, pp. 31–32). The mixed-integer models are usually difficult to solve for small models because of an extremely high number of possible combinations. Thus, a complete enumeration in a reasonable time is not possible (Suhl & Mellouli, 2013, pp. 131–133). If it were possible to solve large problem scenarios fast and optimally, this would be a clear advantage regarding static scenarios with great planning certainty. In dynamic production environments with many disturbances, like short-term changes, machine failure or work loss, plans often have to be recalculated. Hence, the calculated solution with its decisions that have been taken may not necessarily be optimal in retrospect. Therefore, it is important to allow

some flexibility within the plan in order to achieve better overall results (Branke & Mattfeld, 2005, pp. 3105–3107).

Solving a dynamic VRP is formulated as a mixed integer linear programming problem by Haghani & Jung (2005). Chien, Balakrishnan, & Wong (1989) solve the inventory allocation and VRP by using a mixed-integer optimisation. Rieck & Zimmermann (2010) developed a mixed-integer linear model for the VRP with docking constraints.

2.3.2 Heuristic methods

Heuristic methods or heuristics are applied if exact solutions are difficult to determine due to the complexity of the considered system. A heuristic is defined as a search method that tries to achieve an optimal solution in a reasonable time without guaranteeing it. Also, it is not stated how much the calculated solution diverges from the optimum (Heger, 2014, p. 38).

Greedy Randomized Adaptive Search Procedure: Greedy algorithms are algorithms that always decide on the possibility of the largest profit at the current point in time for each decision, but are therefore consequently susceptible to remaining in local optima (Cormen, 2001, p. 370). The Greedy Randomized Adaptive Search Procedure (GRASP), according to Feo & Resende (1988), pursues the approach of randomising these Greedy algorithms in such a way that an optimal global solution can be found.

Two strategies are conceivable. Either several solutions are selected as randomly as possible from the entire solution space, from which the best solution is then selected by the Greedy algorithm. Alternatively, the Greedy algorithm can be used to select the best solutions from the solution space and then randomly select a solution. This procedure can be repeated iteratively, whereby one weak point of the original algorithm is that no learning effect occurs from one iteration to the next. Therefore, variations of the GRASP were developed which integrate this learning process into the algorithm (Prais & Ribeiro, 2000).

For a VRP, GRASP is used by Ferrer, Ortuño, & Tirado (2016) for the coordination of supply after disasters, whereby the target value is the most possible distributed supply. Pino, Martínez, Villanueva, Priore, & Fernández (2012) apply the algorithm to a VRP with time windows in order to increase the number of customers supplied in time. Labadie, Prins, & Prodhon (2016, p. 43) conclude that the solution quality of the GRASP cannot compete with other algorithms, but that there are many approaches in scientific research to combine them with other algorithms.

Particle swarm optimisation: The particle swarm optimisation (PSO) is based on the work of Eberhart & Kennedy (1995) and is modelled on natural swarm behaviour.

Individuals of a swarm follow certain laws in their movements. On the one hand, they try to stay with the group and, consequently follow the group movement. On the other hand they keep a minimum distance from other individuals (Reynolds, 1987, p. 25). This behaviour is applied to the optimisation problem by systematically searching the solution space by single individuals, whose minimum distance significantly reduces the probability of remaining in local minima. Heppner & Grenander (1990, p. 234) create areas within the solution space that are particularly attractive for the individuals of the swarm, so that the swarm is also influenced by the environment. Due to its distribution with many single individuals, the PSO can be easily parallelised and thus the efficiency of the calculation can be increased (Dali & Bouamama, 2015).

The first application of the PSO for VRP originates from Ai & Kachitvichyanukul (2009), who use the algorithm to solve the capacity-limited, route-planning problem. Chen, Hsiao, Himadeep Reddy, & Tiwari (2016) solve an extended VRP using an extended PSO with learning mechanisms. Norouzi, Sadegh-Amalnick, & Alinaghiyan (2015) apply PSO to a VRP with competing routes. In most cases, the PSO achieves good results, but cannot set new standards in computation time (Labadie et al., 2016, p. 91).

Simulated annealing: The SA algorithm was developed by coping with the annealing process of slowly cooling metals to improve their strength by placing the atoms in a perfectly ordered crystal structure with a state of minimal energy. This improvement is only achieved by a suitable cooling rate. The solution search is done within the neighbourhood of the represented problem, typically storing the best solution as well as the previous solution. Best solutions are always accepted immediately by overwriting the previous solution. In case of a poorer solution, the algorithm probabilistically decides whether this deterioration will be accepted or not. At the beginning, the acceptance of deterioration makes it possible to break out of local optima, whereas with increasing iterations, the selection pressure increases. The acceptance threshold is defined as a function of the iteration steps in the so-called cooling schedule. The algorithm terminates after a predetermined number of temperature levels with no improvements (Kirkpatrick, Gelatt, & Vecchi, 1983, p. 672).

Lin, Yu, & Lu (2011) successfully use SA for a VRP under special constraints for loading trailers, finding good solutions in a reasonable time. Wang, Mu, Zhao, & Sutherland (2015)

parallelise the algorithm in order to significantly reduce computing times in the field of pickup and delivery of goods. Recently, the applicability regarding intralogistics has been successfully demonstrated by Pawlewski & Anholcer (2019).

Tabu search: The TS uses a neighbourhood search to find several new solutions. In this way, the best neighbouring solution is selected in each iteration step, even if it is a deterioration. To avoid an endless loop, the last moves are stored in a so-called Tabu list and banned for a fixed period of time. The length of time depends on the length of the Tabu list since each newly added move replaces the oldest move. In this way, it is possible to escape a local optimum (Niehues, 2016a, p. 41). If the algorithm is at a local optimum, all surrounding solutions are worse than the current one. In these cases, the TS selects the solution with the least deterioration. The next step is just the same, because the previous solutions are already on the Tabu list. The algorithm slowly moves away from the local optimum until it is possible to search for new optima again with the standard approach. The TS is usually terminated after a certain number of iterations, or after a maximum number of iterations without improvement on the best solution so far (Stepanenko, 2008, pp. 38–39).

The TS is applied to the VRP in several different variants. Ho & Szeto (2014) apply the TS to a VRP for bike-sharing in big cities. Leung, Zhou, Zhang, & Zheng (2011) formulate a VRP with several additional constraints and introduce a punishment for certain areas of the search space in order to reduce the computation time. Increasing the computational speed is also the goal of Szymon & Dominik (2013), by running multiple TS algorithms at the same time for parallel computation of optimised graphics processing units and thus generating significant advantages of speed. The application examples show that the use within a dynamic system is in principle conceivable regarding computing speed and the solution quality.

Ant-colony-optimisation: The ACO algorithms were inspired by the foraging behaviour of natural ants, who quickly find the shortest path between the food source and their nest by acting together and storing information in the form of pheromones (Danninger, 2012, pp. 57–60). The ACO algorithms themselves were first described by Dorigo & Gambardella (1997). Blum & Sampels (2004) solved the challenge of transferring the natural path optimisation problem to sequencing through the representational form of disjunctive graphs, where virtual ants progressively select the order of operations and evaluate the path by objective functions.

Since the problem modelling is very similar to the VRP where numerous applications of the ACO exist, Gajpal & Abad (2009) and Montemanni, Gambardella, Rizzoli, & Donati (2005)

use the ACO algorithm in each case for dynamic VRPs, where customer orders only arrive during execution. Donati, Montemanni, Casagrande, Rizzoli, & Gambardella (2008) and Balseiro, Loiseau, & Ramonet (2011) describe an implementation of the ACO that takes into account dynamic time slices of the route times. Schyns (2015) considers a dynamic system at an airport, where deliveries by refuelling trucks should always be carried out as quickly as possible in order to remain as responsive as possible. The computation times are within an acceptable range for dynamic problems, especially the approach of Schyns (2015) was explicitly designed for a dynamic problem. Regarding intralogistics, research focuses on coordination and control of means of transport (Micieta et al., 2018b; Teschemacher, 2019).

Genetic algorithms: The GAs are based on the biological principle of evolution. In addition to the mutation and recombination operators, also referred to as genetic operators, the description of a GA also contains components for coding and decoding the solution into the corresponding form of representation, as well as the fitness function, selection strategy and acceptance mechanisms. In addition, the procedure for generating the initial population, consisting of a certain number of chromosomes, and the termination criterion, are part of a GA (Niehues, 2016a, p. 42). Each individual chromosome provides a possible solution to the GA. After creating an initial population by mutation of the starting population, the fitness of each chromosome is calculated and selected for mutation. For the mutation operation, a new solution is created by changing the solution within an individual chromosome. Alternatively, new solutions can also be created by the crossover operation. Thereby, the solutions of two individual chromosomes are combined from which a new chromosome with a new solution emerges. The acceptance mechanism selects which individual chromosomes are taken into the new generation. The new generation will be the initial population for the next iterative step. The new iteration loop begins with the fitness assessment of the new initial population. This process is repeated until a termination condition is met (Niehues, 2016a, p. 42).

By way of example, Haghani & Jung (2005), who model a dynamic VRP with time-dependent route times, provide use cases for a GA in the area of route planning. Kepaptsoglou, Fountas, & Karlaftis (2015) also apply the GA to a dynamic problem using weather forecasts to estimate the speed of container ships. Novaes, Bez, Burin, & Aragão (2015) model and solve a dynamic just-in-time scenario, while the efficiency of the system can be increased by including additional trucks. The use cases show that an application in dynamic systems under certain framework conditions is reasonable.

Hybrid algorithms: The problem-specific heuristics developed to improve the quality of the solution and the computation time are often a combination of different solution methods, so called hybrid algorithms. Hybrid algorithms often lead to better results instead of using only a single algorithm (Jain & Meeran, 1998, p. 37). A multitude of approaches use the genetic algorithm as a meta-strategy integrating problem-specific neighbourhood searches to generate new individuals (Gonçalves, Magalhães Mendes, & Resende, 2005; W. Ho, Ho, Ji, & Lau, 2008; Vidal, Crainic, Gendreau, & Prins, 2013). The use of a population search procedure as a meta-strategy for a neighbourhood search is also referred to as a memetic algorithm (Gao, Zhang, Zhang, & Li, 2011, p. 704).

Yu, Yang, & Yao (2011) use an ACO algorithm in combination with a TS to solve the VRP. Bent & van Hentenryck (2006), in contrast, use an SA algorithm and combine it with a neighbourhood search to create new solutions. Subramanian, Penna, Uchoa, & Ochi (2012) developed a hybrid algorithm to determine the best fleet composition of a heterogeneous fleet. The shifting bottleneck algorithm is also used as a meta-strategy for neighborhood searches, see Scholz-Reiter, Hildebrandt, & Tan (2013).

2.3.3 Performance comparison of the methods

There are numerous performance comparisons of different solution algorithms in scientific research. Hooker (1995) criticises the comparisons based on simulation runs, since, for example, efficient programming strongly influences the quality and runtime of the solution. In addition, the information provided in publications regarding solution quality and computation time is difficult to compare. On the one hand, different hardware was used for the optimisation and on the other hand the performance strongly depends on the software environment and the quality of the implementation (Silberholz, Golden, Gupta, & Wang, 2019, p. 601). Due to the degree of development of the computing power, which depends on the time of publication, it is also difficult to compare the times achieved in various research projects. In addition, according to Hooker (1995), the choice of the example instance has an influence on the development of the algorithm, since these were usually set up for a specific algorithm. In research, disagreement exists about the relationship between the quality of the initial solution, which is absolutely necessary in neighbourhood search procedures, and the solution quality and computation time of the optimisation result (Niehues, 2016a, p. 45).

The general comparison of different solution algorithms is hardly possible due to the above-mentioned criteria, so that in the selection and development of algorithms, the fulfilment of the

problem-specific requirements is relevant. The solution search of these methods is based on a defined target system. Therefore, the logistic target criteria important for this research study are presented in the following section.

2.4 Logistics target criteria

According to the range of tasks, intralogistics includes both production logistics as well as transport logistic objectives. The target system of production logistics can be described in terms of logistics costs and logistics performance. The logistics performance is perceived by the market and can be evaluated on the basis of delivery time and delivery reliability. Short delivery times require short throughput times, while a high delivery reliability requires a high punctuality of the order processing internally. The logistics costs consist of capital commitment costs and production costs. The production costs depend in particular on the utilisation of the production facilities used, while low capital commitment costs require low stocks (Wiendahl, 2010, p. 252).

The different interests of the customers and the company result in an internal conflict of goals, which is also called the dilemma of production control. From the customer's point of view, short lead times and strict adherence to schedules are the overarching goal, so that the ordered products are available as quickly as possible on the agreed delivery time. On the one hand, from a business perspective, capacities should be high and evenly utilised to avoid downtime costs. On the other hand, inventories of raw materials and finished goods should be as low as possible in order to minimise interest on current assets and reduce logistical costs for storage, transport and handling.

In general, there is a shift of the importance from the operational to the market-related target criteria. Delivery reliability and delivery time are in the foreground. Since low stocks are also demanded at the same time, the utilisation inevitably no longer obtains the highest priority (Wiendahl, 2010, p. 253).

This target system, which was originally developed for production logistics, can be transferred to other functional areas of the logistics chain with different target values (Wiendahl, Nyhuis, & Grabe, 2007, pp. 472–473). Based on the objective of logistics service providers in the field of transport logistics, Böse (2012, p. 12) defines a target system for transport logistics based on Wiendahl (2010, p. 253), which can be assessed in terms of logistics performance as the delivery reliability and delivery time and regarding logistics costs as transport costs and capital commitment costs. The associated target values are therefore ‘high adherence to schedules’,

'short order waiting and transportation times', 'high capacity utilisation and transportation frequency', as well as 'minimum empty runs and high flexibility'. Similar to the target system of production logistics, short delivery times require short waiting times and transportation times, while a high delivery reliability requires a high punctuality of internal order processing. The capital commitment costs can be reduced by effectively controlling the means of transport, enabling higher throughput of customer orders, thus reducing inventory costs. In addition, transport costs can be positively influenced by high utilisation of the means of transport.

The target system of intralogistics, consisting of production logistics and transport logistics target figures, is summarised in Figure 2.2, based on Wiendahl (2010, p. 252) and Böse (2012, p. 12).

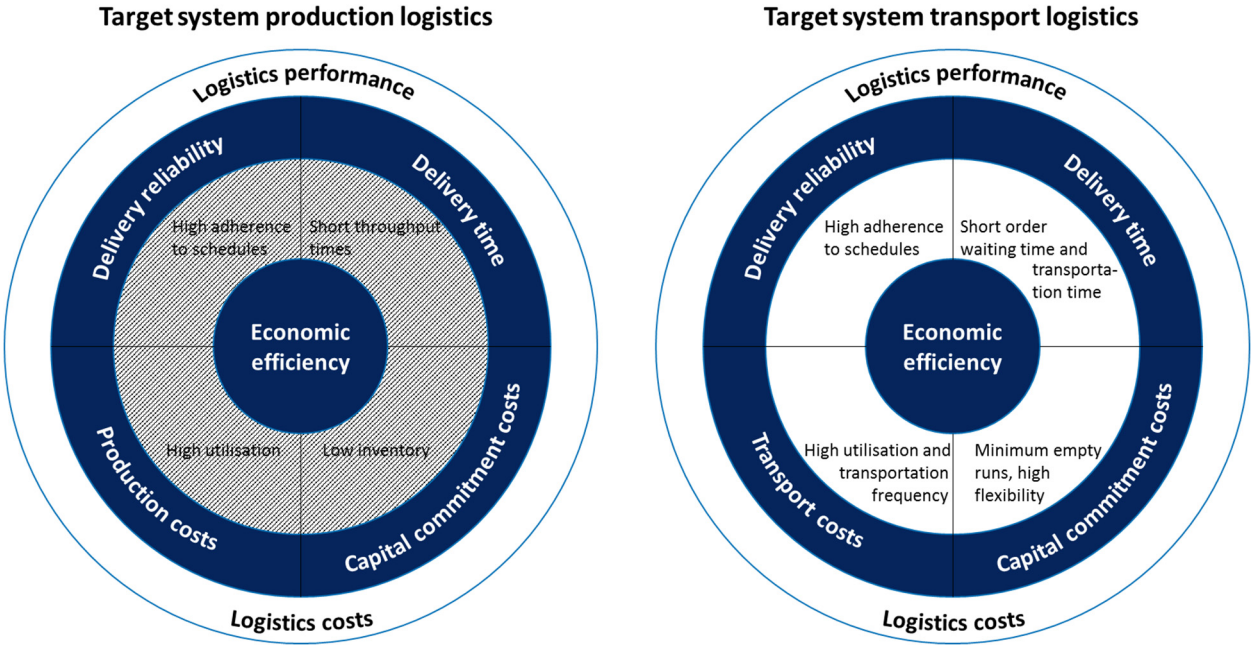


Figure 2.2: Target system of intralogistics based on Wiendahl (2010, p. 252) and Böse (2012, p. 12)

The objective of this thesis is the optimisation of the use of intralogistics means of transport in a flexible flow production. Relevant for the research work in the context of the development of a new control system are thus the target criteria of the target system of transport logistics.

2.5 Werk150, the factory of the ESB Business School

The Werk150, the factory of the ESB Business School on the Reutlingen University campus, represents a realistic production environment to develop and test current issues of applied research, as well as new methods, tools, future technologies and control methods for logistics systems. Thus, the Werk150 serves as a production company, especially focusing on processes

like system engineering, logistics, production and additive manufacturing (Hummel, 2014). The organisational structure of the Werk150 at Reutlingen University can be classified, based on the aspects in Section 2.1, as flexible flow production in this context. For this research study, the production environment of the Werk150 serves as a template for the autonomous system to be developed.

At the Werk150 it is possible to store, assemble, pack, and ship multivariant city scooters. Since the city scooter offers modularity, high variance and reusability, it was chosen as a product for the Werk150. The city scooter consists of approximately 60 single components and is available in different sizes, shapes and colours. For the production, the assembly plant is equipped with lightweight robot systems, autonomous guided vehicles (AGV), communication and information technology as well as additive manufacturing technology (Hummel, Ranz, & Schuhmacher, 2019, p. 351). Appendix A shows the real production environment of the Werk150, an example of the product structure of a city scooter of the model 'FlexBlue' and the considered means of transport in this research study.

2.5.1 Production environment of the Werk150

In this section the production environment of the Werk150 is introduced. The layout of the production environment of the Werk150 is shown in Figure 2.3. The Werk150 consists of a goods receipt, here marked as purple area and a goods issue, shown in turquoise. The goods receipt contains all the A parts, parts with a high-value share and a low-quantity share, which are used to produce a city scooter. In the goods issue, the finished city scooters are stored before the final delivery to the customer. In addition, there is a supermarket for the collaborative tugger train, marked as the yellow area. Here all C parts, parts with a low-value share and a high-quantity share, are located that are needed for the assembly of a city scooter. On the production line, shelves for material supply are available at every workstation. These are shown in Figure 2.3 in light blue boxes.

For the production of one city scooter, the incoming customer order is divided into two; 'stem' and 'base'. The parts of the stem are picked and placed into a fixture at the workstation 'order picking 1', and the parts of the base at the workstation 'order picking 2'. These two workstations provide the correct A parts for the production of a city scooter. At this workstation, a robot and a worker work together.

Then the stem is brought to the workstation WS 2.1 or to the workstation WS 2.2 depending on the workload of the respective workstation. The stem will be delivered to the workstation,

depending on the shortest waiting time for the order. The same process applies in principle for the base. The base is delivered either to workstation WS 1.1, or to workstation WS 1.2. At these workstations, the respective order for the stem or the base is assembled by the workers.

Afterwards, the order is transported to the workstation wedding/packaging. In this assembly step, the stem and the base are assembled into a complete city scooter. This process is described as wedding, similarly to the automotive industry at the point where engine and 'body in white' merge together. Again, two identical workstations are available, which in turn are selected depending on the shortest waiting time for the order.

After the finished city scooter has been packed, it is then brought to the goods issue waiting for shipment to the customer. During the production process, the products are transported on fixtures. After the city scooter has arrived at the goods issue, the fixture is brought back to the beginning of the production, to the two order picking stations.

The green line in Figure 2.3 shows the route of the collaborative tugger train, also called KollRo. The KollRo can only move through the production in one direction. The green stop signs indicate the stops for the tugger train to deliver parts for the city scooter. The orange stop signs indicate the pick-up and drop-off stations of the other means of transport. The other means of transport can move freely through the production. The considered means of transport and their technical requirements are described later in Section 2.5.4.

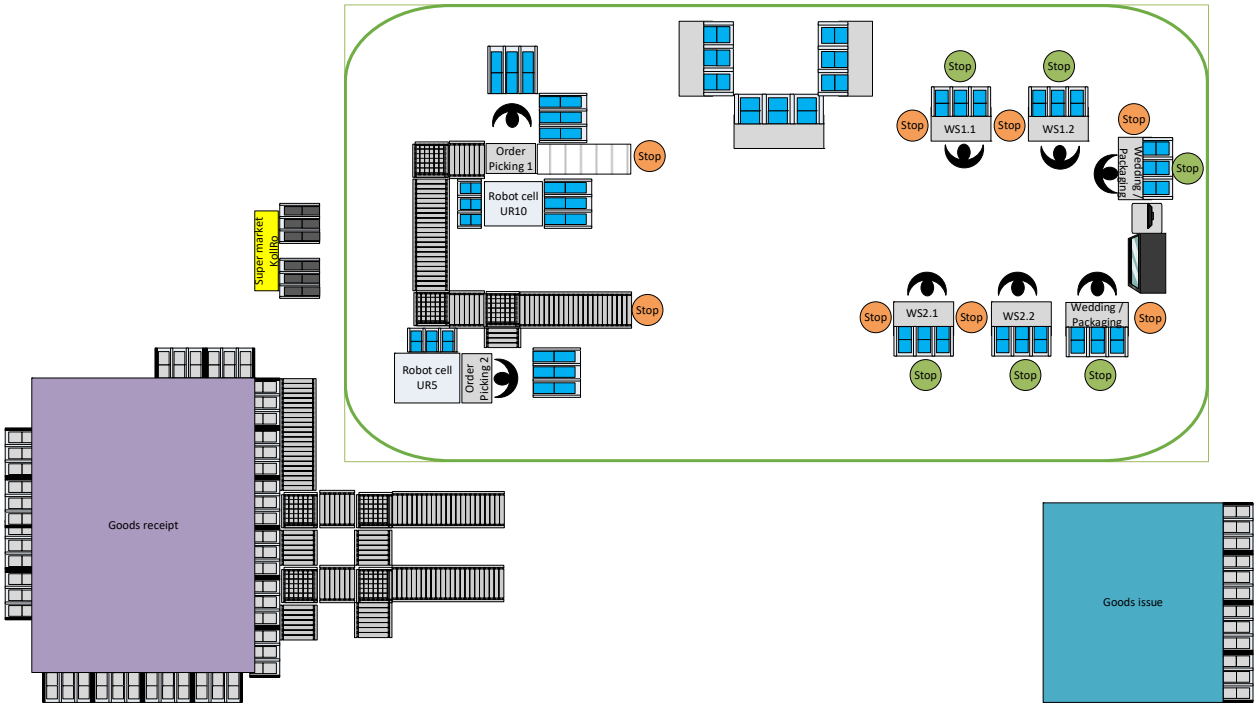


Figure 2.3: Layout of the production environment of the Werk150

2.5.2 Product description of the ‘FlexBlue’ and ‘FlexAir’

The Werk150 is able to produce multivariant city scooters. For this thesis two different models, called ‘FlexBlue’ and ‘FlexAir’, are taken into consideration. The two products differ in some components, but in principle, their assembly steps do not differ. Therefore, the assembly priority plan of the ‘FlexBlue’ and the ‘FlexAir’ are shown below. The assembly priority plan of the ‘FlexBlue’ differs from the ‘FlexAir’ only in the component of the splash guard, framed in red. The assembly priority plan shows the individual components at the respective workstation, regardless of the exact order of the assembly steps, since they are not relevant for this work. The observation period begins at the workstations ‘order picking 1’ and ‘order picking 2’. Figure 2.4 shows the sequence of the workstations an order goes through during production. For each workstation the respective processing time for the assembly of a ‘FlexBlue’ or a ‘FlexAir’ are shown. In addition, the required individual parts are listed at the respective workstation. The number after the parts indicates the required quantity per city scooter.

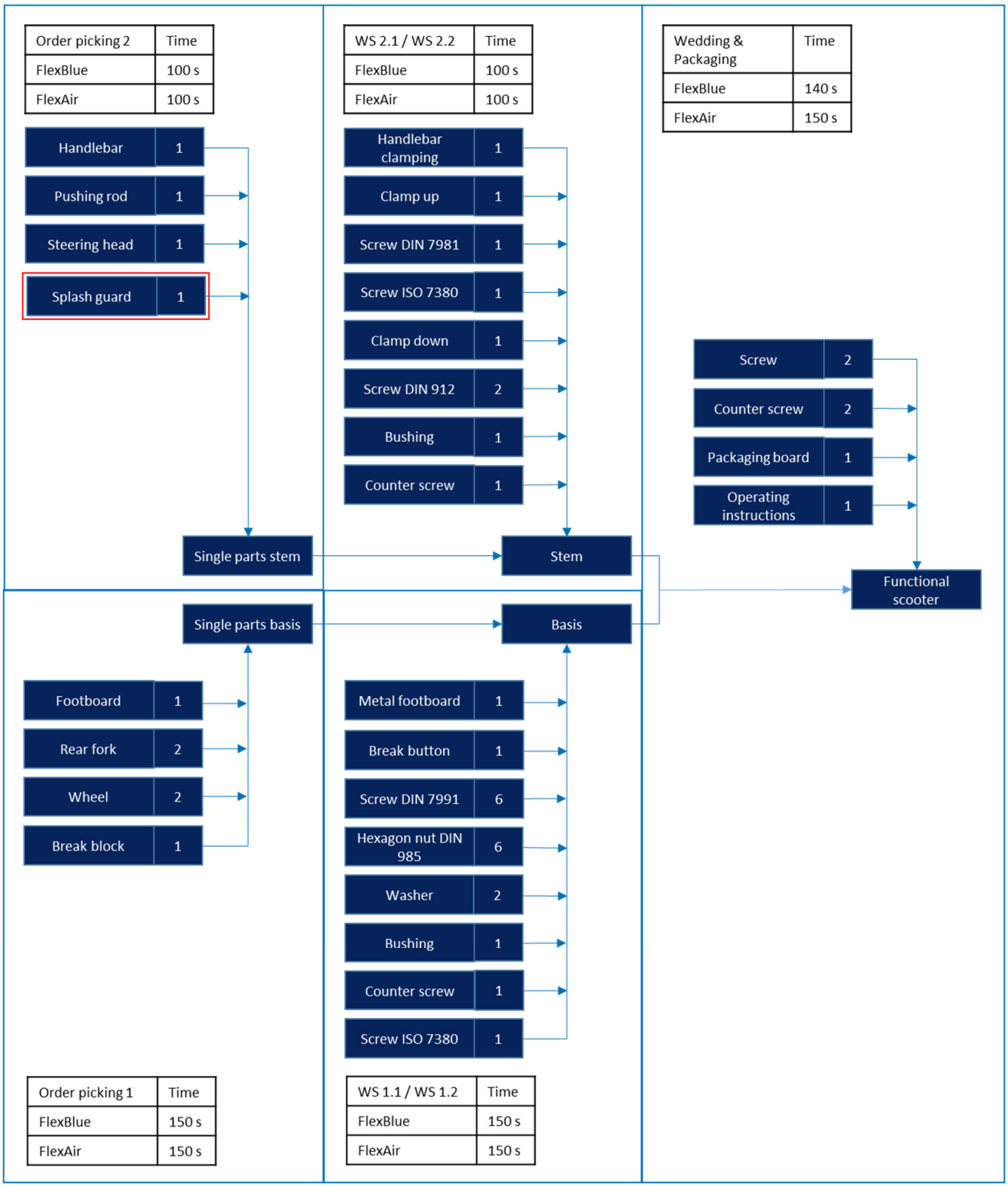


Figure 2.4: Assembly priority plan of the models ‘FlexBlue’ and ‘FlexAir’

2.5.3 List of components of a city scooter

This section categorises each component needed to assemble a city scooter into A and C parts (see Table 2.1). The A parts of the models ‘FlexBlue’ and ‘Flex Air’ differ in the colour of the components and, as already mentioned above, the ‘FlexBlue’ has the splash guard as an additional component, marked in red. The C parts are identical for both city scooters.

Table 2.1: Classification of the components in A and C parts

Component	Value of parts
Handlebar	A
Pushing rod	A
Steering head	A
Splash guard	A
Footboard	A
Rear fork	A
Wheel	A
Break block	A
Handlebar clamping	C
Clamp up	C
Screw DIN 7981	C
Screw ISO 7380	C
Clamp down	C
Screw DIN 912	C
Bushing	C
Counter screw	C
Metal footboard	C
Break button	C
Screw DIN 7991	C
Hexagon nut DIN 985	C
Washer	C
Screw ISO 7380	C
Screw	C
Packaging board	C
Operating instructions	C

2.5.4 Considered means of transport for the developed system

This section describes the considered means of transport for the developed system. For the internal material supply, the Werk150 has two AGVs, the NeoKu with a mobile robot platform, also called Neoku, and the KollRo. In addition, the term ‘means of transport’ also includes the human being in this research study. The means of transport considered were selected because of their different technical properties and degree of flexibility. This increases the complexity of selecting the optimal means of transport for each transport order. In the following, the

technical properties of the means of transport are described. In addition, the means of transport are assigned to specific products and components.

AGV: The Werk150 has two AGVs for the internal material supply. The vehicles consist of a mobile robot platform from the company Neobotix of the model MPO-700. Table 2.2 describes the technical properties of the AGV in terms of load and speed. Since the weight of the individual components as well as the finished city scooter are below the maximum load, this property can be ignored in terms of system development.

Table 2.2: Technical requirements of the AGV (Neobotix, 2019)

Neobotix MPO-700	
Load capacity	400 kg
Speed	< 0.9 m/s

The AGV can transport the individual A parts to the respective workstation as well as the stem, the base or the complete city scooter through the production. The AGV can transport only one transport order at a time. The AGV is not railbound, hence it moves freely through the production.

NeoKu: For unplanned express orders, the NeoKu can be used in the Werk150. The vehicle also consists of a mobile robot platform from the company Neobotix with the model description MPO-700. Additionally, on the mobile robot platform, a lightweight robot from the company Kuka AG is installed. Table 2.3 describes the technical properties of the NeoKu in terms of load and speed.

Table 2.3: Technical requirements of the NeoKu(Kuka AG, 2019; Neobotix, 2019)

Kukaliwa	
Load capacity	14 kg
Speed	< 0.9 m/s

The NeoKu can only transport single A parts, but not the stem, the base or the complete city scooter itself. The NeoKu can only transport one transport order at once. The NeoKu is, like the AGVs, not railbound and can move freely through the production.

KollRo: The KollRo, is the only vehicle responsible for providing the production line with C parts. The KollRo also has a mobile robot platform of the model MPO-700 from the company Neobotix. Several waggons containing C parts are attached to the KollRo. Table 2.4 describes the technical properties of the KollRo in terms of load and speed.

Table 2.4: Technical requirements of the KollRo (Neobotix, 2019)

KollRo	
Load capacity	400 kg
Speed	< 0.9 m/s

The KollRo can only transport C parts to the different workstations, A parts are not assigned to the KollRo. The KollRo can carry all existing C parts at once and is the only means of transport which is railbound. The KollRo can carry two boxes of each C part per tour.

Human: The human is also considered as a means of transport in this thesis. Thereby, the human has the same tasks as the AGVs. The human transports the required individual A parts to the production line and carries the stem, the base or the complete city scooter through the production. The properties of the worker are derived from literary studies. The 'Hettinger table' from Hettinger (1982, p. 96) is used to determine the property load handling and the standard DIN EN ISO 13855 for the property speed (Deutsches Institut für Normung e.V., 2010, p. 15). The 'Hettinger table' suggests a load handling of 10 kg for the human. The derivation of the property load handling for the human is described in detail in Appendix B. Since the restriction for the load handling of the human is 10 kg, the human is allowed to carry one complete city scooter at a time. According to the standard DIN EN ISO 13855, humans reach a speed of 1.6–2 m/s (Deutsches Institut für Normung e.V., 2010, p. 15). For the calculations in the research study and later in the simulation, the average speed of 1.6 m/s is considered for the human. In summary, the properties of the human are shown in Table 2.5.

Table 2.5: Technical requirements of the human

Worker	
Load capacity	10 kg
Speed	< 1.8 m/s

The human is able to transport the single A parts to the respective workstations and carry the stem, the base and the complete city scooter through the production line. The human can transport a maximum of two transport orders at once; in the case of a finished city scooter only one. The human can move freely through the production line.

Chapter 3
Literature review

Chapter 3 describes the literature research based on the process model of Kotzab & Westhaus (2005, p. 94). The demarcation of the literature search was already mentioned in Section 1.4.1.2. Therefore, this section starts with a categorisation of the observed literature. Criteria were defined for the examination of the considered literature followed by an evaluation. The results of the literature research are then summarised. Finally, the research gap is derived from the results of the literature review.

3.1 Structured content analysis

The following categories have been used for a structured content analysis. The categories are divided into the keywords ‘intralogistics’ and ‘autonomous control’, which form the basis of this work. Furthermore, the terms ‘transportation control’, ‘transportation logistics’, ‘optimisation’ and ‘simulation’ were taken into consideration. The literature was examined for the following criteria and the degree of consideration was defined:

Multi-objectives: The scientific work is examined regarding their integration of logistics target criteria. In this context, attention will be paid to whether only a single target figure, such as throughput time or several target criteria are considered. Research studies focusing only on a single target figure do not consider this aspect; research studies applying multi-objective fulfil this aspect completely.

Mixed transport resources: The research studies were analysed with regard to a holistic view of the logistics system. In this thesis, the term ‘holistic view’ means that the research studies consider different means of transport. A distinction is made as to whether the research area focuses on a single means of transport, means of transport of the same type or different means of transport with specific properties.

Error handling: The developed systems in the examined research studies are assessed with regard to their flexibility. During the literature review, special attention was paid to the extent to which the analysed research studies took into consideration unpredictable disturbances, changes, etc. and subsequent measures to counteract these dynamics. The fulfilment of the criterion is subdivided as follows: error-free environment, handling errors before or after the system execution, and error handling during the system execution.

Optimisation: This aspect considers scientific studies regarding the behaviour of the system with respect to its target system. The systems are assessed as to what extent the system can adapt to their target figures in order to ensure a high degree of target achievement. The time of occurrence of the optimisation and the frequency of the optimisation process in order to achieve an up-to-date plan are also considered. The systems are differentiated depending on the time of the optimisation. The fulfilment of the criterion is subdivided as follows: no optimisation, optimisation before or after the system execution and optimisation during the system execution.

Method of sequence planning: The various methods used for sequence planning for the control of intralogistics means of transport were analysed and compared. The various methods and their capabilities were examined regarding their suitability for this research study. Since it is difficult to compare the performance of a method for sequence planning (see also Section 2.3.3), this criterion is differentiated according to the type of decision-making. The system can make its decisions on a centralised, decentralised, or hybrid basis.

3.2 Summary and evaluation

Based on the objectives of this thesis, the approaches of intralogistics control of means of transport were analysed regarding their suitability for a flexible flow production. The approaches were observed with regard to their target criteria and examined according to the used means of transport and their holistic perspective. The suitability as a control method in terms of the disturbance reaction was assessed, the optimisation of the solution space was analysed and finally the selected control architecture with respect to their computation time was characterised. A summary of the most important research studies regarding control of intralogistics means of transport, considering only publications from 2016 onwards, is presented in Table 3.1.

Over the past decades, a number of algorithms have been developed and improved addressing the problem of sequencing in intralogistics including optimising and heuristically procedures that can solve specific problems optimally in relation to the selected target criteria. In literature,

several articles tackled the problem of solving an optimised target achievement, considering production and transport aspects (Burduk & Musial, 2016). Although Zawisza (2018) considers several hierarchical target structures in his work, he neglects heterogeneous target structures. In contrast, Kang & Bhatti (2019) present a framework where throughput time and total inventory holding cost are used as two combinatorial optimisation objectives.

With regard to the selection of the considered means of transport in the various research studies, it is noticeable that in the literature only one type of means of transport is considered for the system to be developed. Often, the AGV is the object of reflection as a means of transport in the papers (Bochmann, 2018; Grzegorz Kłosowski, Arkadiusz Gola, & Thibbotuwawa Amila, 2018; Micieta et al., 2018a). The AGV is capable of localising itself in a known environment and finds its path through a discrete bidirectional topological graph (Walenta, Schellekens, Ferrein, & Schiffer, 2017, p. 1441). In addition, research is focused on the optimisation of milk runs. The aim of the research studies is to improve the route-finding of tugger trains (Emde & Gendreau, 2017; K. Pawlewski & Anholcer, 2019; Teschemacher & Reinhart, 2017). Kousi, Michalos, Makris, & Chryssolouris (2016) and Grijalva, Chávez, & Camacho (2018) do not link their work to a specific means of transport, but the system developed is limited to one means of transport and cannot be applied to different ones, e.g. tugger train or AGV. In addition, observing the current solutions, it is striking that usually vehicles only make a single tour, recurring vehicles are not considered for transport.

Through the high degree of dependencies in flexible flow production, the effects of disturbances are critical. Delays in part supply or assembly steps directly influence other orders, as incomplete steps or missing parts can lead to a greater amount of rework. In dynamic systems, these disturbances should ideally be reduced to a minimum by carrying out specific measures. For example, Greenwood (2016) implements planned and unplanned downtimes, such as operator break, battery recharging, and malfunction requiring repair. However, these downtimes only occur after the fulfilment of the transport task; thus recalculation of the solution is not considered. Kłosowski, Gola, & Amila (2018) assesses the risk that due to lack of material, the production line is forced to stop.

The analysis of the state-of-the-art research reveals, with regard to optimisation of systems in terms of external changes, that existing approaches can only partially solve these challenges. For example, only a few algorithms can handle disturbances or order changes. In addition, they are usually designed for static benchmark instances and are thus not optimised for dynamic response capability (Kalayci & Kaya, 2016; Zenker, Emde, & Boysen, 2016). Therefore, the

generated process cannot be met over a long period of time due to the uncertainties and thus loses its validity (Bochmann, 2018, pp. 163–164). In contrast, Kang & Bhatti (2019) use a modified genetic algorithm and discrete event simulation to achieve multi-objective optimisation. Blesing, Luensch, Stenzel, & Korth (2017, p. 26) apply the fuzzy logic algorithm to influence the decision-making process in order to optimise the target achievement.

Fundamentally, centralised procedures have all the information they need to consider when calculating an overall schedule, e.g. Kang & Bhatti (2019) or Gotthardt, Hulla, Eder, Karre, & Ramsauer (2019). The disadvantage of the optimising methods and the central heuristics is that they usually have to calculate overall plans and perform a computation-intensive recalculation when changes occur. Li, Liu, Xiao, Yu, & Zhang (2017) state that centralised approaches deserve attention in applications that are sensitive to solution quality but insensitive to computation time. To meet the requirements, in particular with regard to the dynamic aspect, decentralised approaches are needed that can react to changes with the help of local information and still ensure a good achievement of the target criteria.

Heuristic methods reduce complexity by limiting the solution space and, for example, creating solutions from partial solutions or improving previously found solutions by minor changes. In Blesing et al. (2017, p. 26) and Sohrt et al. (2017) agents include previous solutions to reduce the computation time of the new calculation. In the system of Teschemacher & Reinhart (2017, pp. 764–765), two algorithms are calculated simultaneously to find the best option. This enables them to generate solutions much more quickly, but at the same time, they accept a reduction in the quality of the solution. What is of interest is not only the solution quality but also the associated computation time. Arora & Arora (2016, p. 224) compare a GA with an ACO algorithm to solve the TSP. The GA provides a better result in terms of distance travelled, but the computation time of the ACO algorithm is better than that of the GA. Silva, Sousa, & Runkler (2008, p. 351) conclude that GA and ACO both have a good and similar performance in terms of different problems, although the GA requires less computational time than the ACO. However, the ACO uses the extra time to record information about the optimisation procedure that can be used to reschedule the logistics system in dynamic environments. Chmait & Challita (2013, p. 222) and Musa & Chen (2008, p. 849) both compare the SA with an ACO for scheduling problems. According to Chmait & Challita (2013, p. 222) SA achieves better results in terms of finding the solution. However, the ACO needs a shorter computing time than the SA algorithm. In contrast, the outcome of the findings of Musa & Chen (2008, p. 849) presents that the ACO outperforms the SA algorithm. Mukhairez & Maghari (2015, p. 653) present a

comparative study of an ACO, a GA and a SA in terms of shortest distance and execution time. The results of the study show that the SA algorithm has the shortest computation time and the ACO achieved the best results regarding shortest distance between cities. Another paper comparing these three algorithms carried out by Kumbharana & Pandey (2013, p. 228) concludes that the GA provides better solutions more often than the ACO algorithm or the SA algorithm.

Recent work has mainly applied GAs with Embedded Neighbourhood Search, which is assigned to hybrid algorithms, achieving very good results, inter alia, Gonçalves et al. (2005) and Gao et al. (2011). Kalayci & Kaya (2016) use a hybrid algorithm to solve the VRP. Comparing the developed algorithm with benchmark problem instances show that in general this algorithm outperforms an ACO with embedded local search strategies. Furthermore, the proposed algorithm performed well in terms of the solution quality and the computation time compared to the algorithms in literature. Gola & Kłosowski (2018) and Kłosowski et al. (2018) show the effectiveness of hybrid algorithms for optimising pathfinding for large AGV systems. In summary, it can be concluded that hybrid algorithms achieve good solutions, in particular by a combination of genetic algorithms and neighbourhood search methods.

Table 3.1 shows a summary of the research studies considered with regard to the fulfilment of the defined aspects from the structured content analysis. The degree of fulfilment of the respective aspects is shown by ‘Harvey balls’.

Table 3.1: Summary of selected publications regarding the control of intralogistics systems

References	Literature observed with regard to following criteria				
	Multi-objectives	Mixed transport ressources	Error handling	Optimisation	Method of sequence planning
Blesing et. al (2017)	●	◐	○	●	D
Bochmann (2018)	●	○	◐	●	D
Burduk and Musial (2016)	●	○	○	●	C
Emde and Gendreau (2017)	●	◐	○	◐	C
Gola and Klosowski (2018)	●	○	○	◐	H
Gotthardt et. al (2019)	●	○	○	◐	C
Greenwood (2016)	○	◐	●	○	D
Grijalva et. al (2018)	●	◐	○	◐	C
Kalayci and Kaya (2016)	○	○	○	◐	H
Kang and Bhatti (2019)	●	○	●	●	C
Klosowski et. al (2018)	○	○	●	●	H
Kousi et. al (2016)	●	○	○	○	C
Li et. al (2017)	○	◐	○	◐	C
Micieta et. al (2018)	○	◐	●	○	D
Pawlewski and Anholcer (2019)	○	○	○	●	H
Shchekutin et. al (2017)	●	○	○	●	D
Sohrt et. al (2017)	○	○	◐	●	D
Teschemacher and Reinhart (2017)	○	○	○	●	D
Teschemacher (2019)	○	○	●	●	D
Walenta et. al (2017)	○	◐	○	●	D
Zawisza (2018)	●	○	●	○	D
Zenker et. al (2016)	●	○	○	◐	C

○	Does not consider aspect	C	Centralized
◐	Consider aspect partially	D	Decentralized
●	Consider aspect completely	H	Hybrid

3.3 Research gap

According to Table 3.1, no analysed research study considers all defined aspects. However, the large number of studies provides a good basis for the development of a system for controlling intralogistics means of transport in a flexible flow production in order to adaptively and holistically react to disturbances, even though there is a need for improvement in the areas listed below.

As the analysis of the control methods shows, the suitability of optimising control procedures decreases with increasing heterogeneity of the material flows and work schedules. At the same time, with increasing complexity, the importance of a central overview of order progress, delays and predictable completion times increases, which heuristic concepts cannot achieve either. For this reason, hybrid control concepts are preferred for the flexible flow production of highly individualised products. There is a need for improvement in developing a suitable control method that enables a short-term, autonomous reaction to disturbances considering the best possible solution.

An extensive search of the solution space is achieved by recalculating the order sequence with optimisation algorithms, where iterative heuristics are of particular importance. The challenge of the optimal solution led to a multitude of solution algorithms. It is noticeable that the studies are mostly limited to theoretical problem solving. The practical application regarding the control of means of transport in intralogistics is thus considered only rudimentarily. Even large VRPs can be solved optimally, or almost optimally, in a short time with a multitude of optimisation methods. The highest potential is offered by the use of metaheuristics with embedded neighbourhood search, especially GAs. The concrete design of these algorithms is very important since the integration of boundary conditions of real production scenarios increases the computation time. Depending on the control architecture, there is a need for improvement in the development of a sequence planning algorithm that solves realistic VRP with all existing boundary conditions in the best possible way and in a short time.

Error handling offers the potential to make specific adjustments on a small scale, without new production planning, by initiating measures depending on disturbances. There is still a need for action in the integration of possible disturbances into the optimisation procedures, which includes all possible disturbances from reality, which affect the production process and enable a derivation of measures. Based on this, measures are to be developed which are executed accordingly in order to ensure the highest possible target achievement.

Finally, every measure or re-sequencing requires the evaluation of good results. One challenge is the consideration of different target figures and regarding measures, the possibility that the action undertaken leads to the opposite effect. Taking different target figures into account promises great potential in order to quantify and calculate them against each other.

Chapter 4
Problem definition

The chapter first contains a description of the problem definition of the simulation study. The questions to be investigated and the objectives to be achieved are defined. Thereby, the control processes are examined with regard to their weaknesses. Subsequently, the control processes of the autonomous control system to be developed are described. Thus, a target system is defined, which contains target figures characterising the behaviour of the logistics system of the Werk150 with regard to its tasks. The target system distinguishes between the transport of the city scooter and A parts by the means of transport AGV, NeoKu and human, and the supply of the production line with C parts by the KollRo.

4.1 Problem definition and objectives of the simulation study

The subject of the simulation study is the control of means of transport in a flexible flow production taking the Werk150 as a model reference. The task of the simulation study is to test and quantify the potential for improvement in the area of intralogistics control of the means of transport for order processing through the introduction of autonomous control of logistics transport processes. For this purpose, the existing, central control of the intralogistics means of transport is compared with the conceptually developed autonomous control system. The following thesis will be tested with the help of the simulation study:

> *The implementation of autonomous processes for the control of intralogistics means of transport in logistical order processing ensures a higher degree of target achievement with regard to the logistics target criteria performance and costs than static, central controlled logistics systems.*

The formulated thesis will be examined within subsequent sections. For this purpose, the existing static and central control of the means of transport in the Werk150 is compared with an autonomous control approach using a hybrid decision-making method. The problem underlying the simulation study was described in detail in Section 1.2 and will only be briefly summarised here with regard to the control of intralogistics means of transport in the Werk150.

The existing static and central controlled allocation of transport orders to the various means of transport is based on a predetermined plan with fixed allocation of the transport orders to the

means of transport. The means of transport are assigned to predefined areas and routes within the production for transportation. This means that the system cannot react to unpredictable events, such as assembly errors at the workstations or the failure of a means of transport during transport. This means that in case of unpredictable events, some means of transport may be fully utilised or even overloaded, while other means of transport may have to execute only few or no transport orders at all. In case a means of transport fails during execution, these transport orders can only be executed after the means of transport has been repaired. As a result, the subsequent processes are no longer supplied with the required parts. This can ultimately lead to long throughput times, which in turn has a negative effect on the adherence to schedules of the orders.

In order to counteract the aforementioned weaknesses with regard to the logistics target criteria ‘short order waiting time and transportation time’ and ‘high adherence to schedules’ of the system, an algorithm allocating the transport orders to different means of transport taking the defined logistics target criteria into account was developed (cf. Chapter 6). The algorithm allocates the incoming transport orders to the different means of transport. The means of transport most suitable for the transport with regard to the logistical target criteria receives the transport order. The order allocation to the means of transport is real-time capable and reacts on occurring failures immediately. There is no fixed allocation of transport orders to means of transport, as in the existing system.

The existing static and central control of the means of transport is based on a predetermined processing sequence of the transport orders. Due to the lack of flexibility in terms of disturbances, the fixed processing sequence of the means of transport leads to bottlenecks at the workstations and high waiting times. The high waiting times are reflected in high throughput times, which in turn result in low adherence to schedules of the orders. In addition, the fixed processing sequence of the means of transport results in suboptimal utilisation times of the means of transport, since it is not possible to distribute the orders equally to the means of transport.

The algorithm in Section 6.4 was extended to eliminate the weaknesses described above in terms of ‘low utilisation times of the means of transport’, a ‘high proportion of empty runs’, ‘long waiting and transportation times’ and ‘low adherence to schedules’. The algorithmic control of the order allocation, the selection of the means of transport, takes place dynamically at runtime and thus immediately before the start of the transport process. The selection process is based on the properties of the respective means of transport and their current state.

To be able to compare the existing system with the developed autonomous system, logistics target criteria for intralogistics control of means of transport are defined in the following section.

4.2 Logistics target criteria of the Werk150

On the basis of the described target system for transport logistics in Section 2.4, the target system for the operative control of intralogistics of the production environment of the Werk150 was extended. The target system for the Werk150 distinguishes between the means of transport; AGV, human and NeoKu, which are responsible for the transport of the city scooters, the stem, the base and the A parts, and the target system of the KollRo, which supplies the production line with C parts. The transport logistics subgoals of the intralogistics system are shown in Figure 4.1.

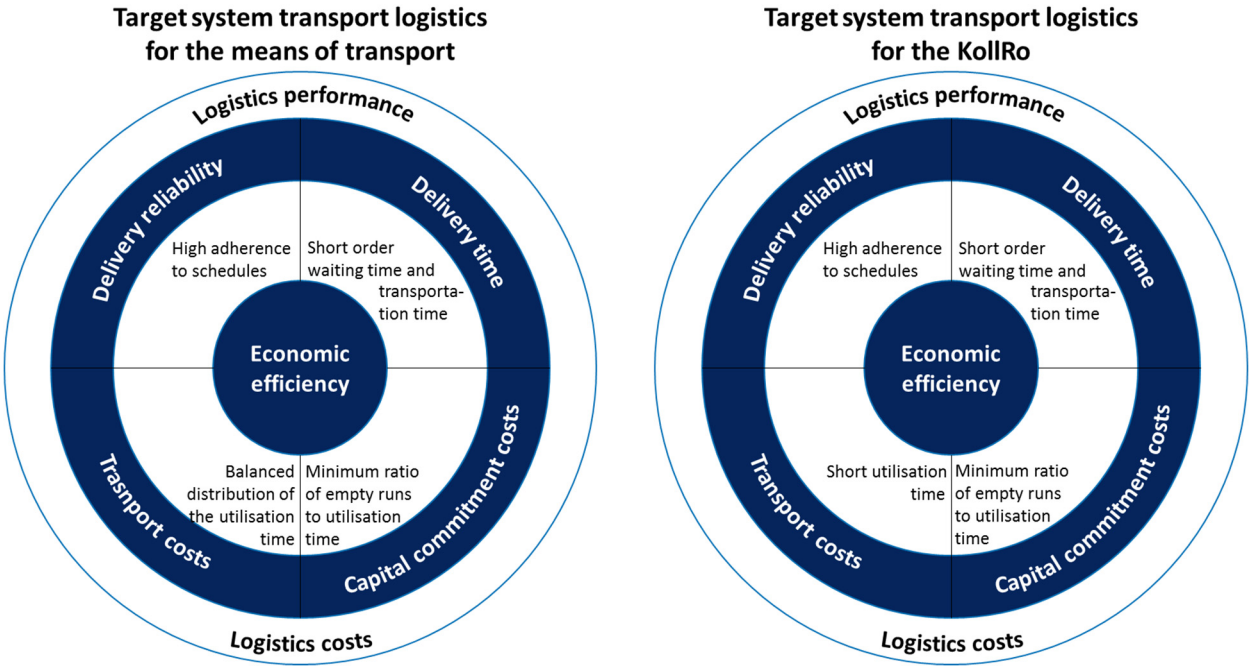


Figure 4.1: Relevant target figure of the simulation based on Wiendahl (2010, p. 252) and Böse (2012, p. 12)

In order to be able to determine the target achievement of the research objectives, suitable target figures are required for measuring the logistical performance of the system. Target figures are the basis to assess and improve the performance of logistics systems (Wiendahl, 2008, p. 363). The logistics target figures for measuring the target achievement of intralogistics systems of the Werk150 are described in the following sections.

As already mentioned, two target systems are considered for the Werk150. One target system considers the transport order for the city scooter, the stem, the base and the corresponding A

parts, referred to as ‘target system for the means of transport’, the other target system considers the supply of the production line with C parts, named ‘target system for the KollRo’. The reason for the distinction between the two target systems is that two different transport systems, which work independently, are considered for these tasks. The transport of the city scooter, the stem, the base and the A parts is carried out by the means of transport AGV, NeoKu and human. For the transport of the C parts to the production line, only the KollRo is responsible. In principle, the target criteria of the logistics performance of the two target systems do not differ, only the type of orders is different. The target system of the means of transport considers the waiting times and transport times as well as the delivery reliability of the city scooter, the stem, the base and the A parts, the target system of the KollRo for the C parts. With respect to transport costs, the target criteria of the two target systems differ. Regarding the target system of the means of transport, the transport orders should be distributed equally among the means of transport. The target figure was modified in comparison with the target system presented in Section 2.4. The system cannot influence the utilisation time of the means of transport because the number of means of transport is fixed and the execution of the transport orders is order-dependent, hence the means of transport waits until it receives a transport order. Therefore, an equal distribution of the utilisation times of the means of transport in the system is desired. In the existing logistics system, the KollRo runs its tour at certain intervals, i.e. not order-related. Hence, the target system of the KollRo aims to reduce the utilisation time. With regard to the capital commitment costs, the two target systems are identical. The aim is to reduce the empty runs by dynamic route planning of the means of transport and flexible adjustments of the system to unpredictable events.

In the following sections, the calculations of the logistics target figures are described using the target system of the means of transport. The calculation of the target figures of the target system of the KollRo are not shown explicitly, since they can be calculated using the same formulas.

4.2.1 Short order waiting time and transportation time

The aim of the system is to minimise the throughput time of the customer orders. The measurement of the throughput time starts with the point of time a customer order of a new city scooter is received until the point of time it is delivered to the goods issue. The throughput time of a city scooter consists of different assigned times:

- Transportation times between goods receipt, goods issue, and workstations
- Waiting times at goods receipt and in front of each workstation
- Processing times at each workstation

The throughput time of a city scooter consists of the transportation, waiting, processing and failure time. Set-up times usually do not occur at the workstations and are therefore not considered further here. The throughput time of a city scooter covers the entire period of the transportation of the city scooter through the production from goods receipt to goods issue.

$$TPT = TT_{total} + TW_{total} + TP_{total} + TF_{total} \tag{4.1}$$

with:
- TPT total throughput time of the city scooter [s]
- TT_{total} total transportation time of the city scooter [s]
- TW_{total} total waiting time of the city scooter [s]
- TP_{total} total processing time of the city scooter [s]
- TF_{total} total failure time of the means of transport [s]

The total transport time of a city scooter consists of the transport time from goods receipt to the workstation picking order 1/2, then to workstation WS 1.1/1.2 or WS 2.1/2.2, then to the workstation wedding/packaging 1/2 and last to goods issue.

$$TT_{total} = TT_{PO} + TT_{WS} + TT_{WP} + TT_{GI} \tag{4.2}$$

with:
- TT_{total} total transportation time of the city scooter [s]
- TT_{PO} total transportation time to the workstation picking order 1 / 2 [s]
- TT_{WS} total transportation time to the workstation WS 1.1/1.2 or WS 2.1/2.2 [s]
- TT_{WP} total transportation time to the workstation wedding/packaging [s]
- TT_{GI} total transportation time to the goods issue [s]

The total transport time of a city scooter to the workstation picking order 1/2 consists of the sum of the transport times from the goods receipt to the respective workstation picking order 1/2. If necessary, the workstation picking order 1/2 can be approached several times in the event of a transport failure.

$$TT_{PO} = \sum_{0}^{q} TT_{PO,x} \tag{4.3}$$

with: TT_{PO} total transportation time to the picking order 1/2 [s]
$TT_{PO,x}$ transportation time from the goods receipt to the picking order x [s]
q number of transport tours

The total transport time of a city scooter to the workstations WS 1.1/1.2 or WS 2.1/2.2 consists of the sum of the transport times from the workstation picking order 1/2 to the respective workstation WS 1.1/1.2 or WS 2.1/2.2. If necessary, the workstation WS 1.1/1.2 or WS 2.1/2.2 can be approached several times in the event of a transport failure.

$$TT_{WS} = \sum_{0}^{q} TT_{WS,xy} \tag{4.4}$$

with: TT_{WS} total transportation time to the workstation 1.1/1.2 or 2.1 / 2.2 [s]
$TT_{WS,xy}$ transportation time from the workstation picking order x to the workstation WS y [s]
q number of transport tours

The total transport time of a city scooter to the workstation wedding/packaging 1/2 consists of the sum of the transport times from the workstations WS 1.1/1.2 or WS 2.1/2.2 to the respective workstation wedding/packaging 1/2. If necessary, the workstation wedding/packaging 1/2 can be approached several times in the event of a transport failure.

$$TT_{WP} = \sum_{0}^{q} TT_{WP,xy} \tag{4.5}$$

with: TT_{WP} total transportation time to the workstation wedding/packaging 1/2 [s]
$TT_{WP,xy}$ transportation time from the workstation WS x to the workstation wedding/packaging y [s]
q number of transport tours

The total transport time of a city scooter to the goods issue consists of the sum of the transport times from the workstation wedding/packaging 1/2 to the goods issue. If necessary, the goods issue can be approached several times in the event of a transport failure.

$$TT_{GI} = \sum_{0}^{q} TT_{GI,x} \tag{4.6}$$

with: TT_{GI} total transportation time to the goods issue [s]

$TT_{GI,\,x}$ transportation time from the workstation wedding/packaging x to the goods issue [s]

q number of transport tours

The total waiting time of a city scooter in production consists of the waiting time at goods receipt, the waiting times in front of the workstations WS 1.1/1.2 or WS 2.1/2.2 and the waiting times after the workstations WS 1.1/1.2 or WS 2.1/2.2.

$$TW_{total} = TW_{GR} + TW_{RP} + TW_{DP} \tag{4.7}$$

with: TW_{total} total waiting time of the scooter [s]

TW_{GR} total waiting time at goods receipt [s]

TW_{RP} total waiting time in front of the workstation WS 1.1/1.2 or WS 2.1/2.2 and wedding/packaging 1/2 [s]

TW_{DP} total waiting time after the workstation WS 1.1/1.2 or WS 2.1/2.2 and wedding/packaging 1/2 [s]

After the system receives an order for a new city scooter, waiting times may occur at the goods receipt or at the workstations WS 1.1/1.2 or WS 2.1/2.2 if the city scooter cannot be transported directly to the next workstation. This is the case, for example, if all means of transport are already occupied with a transport order.

The processing time of a city scooter in production consists of the sum of the processing times at the workstations WS 1.1/1.2 or WS 2.1/2.2. The processing time results from the processing times for the corresponding city scooter model. Additionally, in case of a defective or missing part, the time for ordering and delivery of a new part is added to the processing time.

$$TP_{total} = \sum_{0}^{w} TP_{WS,xj} + \sum_{0}^{r} DP_{total} \tag{4.8}$$

with: TP_{total} total processing time of the city scooter [s]

$TP_{WS,\,xj}$ processing time at workstation WS x for model j [s]

DP_{total} time for ordering the defective or missing part [s]

w number of workstations

r number of defective or missing parts

In case of a failure of the means of transport during transport, either the city scooter has to wait for another means of transport to get picked up or wait until the means of transport is repaired. If the city scooter waits for the means of transport to be repaired, the time for repair is added to the total throughput time of the city scooter.

Therefore, the total failure time of the city scooter consists of the amount of repair time of its means of transport.

$$TF_{total} = \sum_{0}^{f} TR_{MOT,m} \tag{4.9}$$

with: TF_{total} total failure time of the city scooter [s]
$TR_{MOT,\,m}$ repair time of the means of transport m [s]
f number of failures

4.2.2 High adherence to schedules

In intralogistics systems, adherence to delivery schedules is of great importance. In times of just-in-time production, meeting promised delivery times is of ultimate importance for the customer. If the products cannot be delivered in the desired time, the company can be penalised in the form of a fine. If the products are produced before the planned delivery time, the cost of warehousing increases. Therefore, the aim is to meet the desired target time of the customer as precisely as possible.

The adherence to delivery schedules of the customer orders of the city scooters is calculated as the difference between the desired delivery time and the throughput time of the city scooter through the production.

$$DD_{total} = \sum_{0}^{n} DD_C - TPT \tag{4.10}$$

with: DD_{total} total adherence to delivery schedules of the city scooter [s]
DD_C desired delivery time of the customer [s]
TPT total throughput time of the city scooter [s]
n number of completed customer orders

The number of considered customer orders includes all completed customer orders during the period under review. The considered customer orders do not have to be the same as the total number of customer orders in the observed period. The total number of customer orders also includes all customer orders that have not yet been completed by the end of the period.

4.2.3 Balanced distribution of the utilisation time of the means of transport

With regard to the means of transport, the aim is to achieve a balanced utilisation time of the capacities in order to avoid any downtime costs. The utilisation time of the means of transport is a percentage of the total utilisation time of the means of transport in relation to the period under consideration.

$$UT_{total} = \frac{\sum UT_{MOT}}{u} \times 100 \tag{4.11}$$

with: UT_{total} utilisation time of the means of transport [%]
UT_{MOT} total utilisation time of the means of transport [s]
u time of the observed period [s]

The utilisation time of a means of transport consists of the transport time from goods receipt to picking order 1/2, then to workstation WS 1.1/1.2 or workstation WS 2.1/2.2, afterwards to the workstation wedding/packaging 1/2 and finally to goods issue. For the target system of the KollRo only the total utilisation time is of interest.

$$UT_{MOT} = UT_{PO} + UT_{WS} + UT_{WP} + UT_{GI} \tag{4.12}$$

with: UT_{MOT} total utilisation time of the means of transport [s]
UT_{PO} total utilisation time to the workstation picking order 1/2 [s]
UT_{WS} total utilisation time to the workstation WS 1.1/1.2 or WS 2.1/2.2 [s]
UT_{WP} total utilisation time to the workstation wedding/packaging 1/2 [s]
UT_{GI} total utilisation time to the goods issue [s]

The total utilisation time of a city scooter to the workstation picking order 1/2 consists of the sum of the transport times from the goods receipt to the respective workstation picking order 1/2.

$$UT_{PO} = \sum_{0}^{q} UT_{PO,x} \tag{4.13}$$

with: UT_{PO} total utilisation time to the workstation picking order 1/2 [s]
$UT_{PO,x}$ transportation time from the goods receipt to the workstation picking order x [s]
q number of transport tours

The total utilisation time of a city scooter to the workstation WS 1.1/1.2 or WS 2.1/2.2 consists of the sum of the transport times from the workstation picking order 1/2 to the respective workstation WS 1.1/1.2 or WS 2.1/2.2.

$$UT_{WS} = \sum_{0}^{q} UT_{WS,xy} \tag{4.14}$$

with: UT_{WS} total utilisation time to the workstation WS 1.1/1.2 or WS 2.1/2.2 [s]
$UT_{WS,\,xy}$ transportation time from the workstation picking order x to the workstation WS y [s]
q number of transport tours

The total utilisation time of a city scooter to the workstation wedding/packaging 1/2 consists of the sum of the transport times from the workstations WS 1.1/1.2 or WS 2.1/2.2 to the respective workstation wedding/packaging 1/2.

$$UT_{WP} = \sum_{0}^{q} UT_{WP,xy} \tag{4.15}$$

with: UT_{WP} total utilisation time to the workstation wedding/packaging 1/2 [s]
$UT_{WP,\,xy}$ transportation time from the workstation WS x to the workstation wedding/packaging y [s]
q number of transport tours

The total utilisation time of a city scooter to the goods issue consists of the sum of the transport times from the workstation wedding/packaging 1/2 to the goods issue.

$$UT_{GI} = \sum_{0}^{q} UT_{GI,x} \tag{4.16}$$

with: UT_{GI} total utilisation time to the workstation wedding/packaging 1/2 [s]
$UT_{GI,\,x}$ transportation time from the workstation wedding/packaging x to the goods issue [s]
q number of transport tours

4.2.4 Minimum ratio of empty runs to runtime

From the company's point of view, the minimisation of the ratio of empty runs to utilisation time is desirable. By minimising empty runs, a higher efficiency of the control of means of transport can be achieved. Therefore, means of transport should be capable of accepting transport orders during runtime. Otherwise, the system has the disadvantage that, on the one hand, the means of transport is not available for any new transport orders during the runtime and on the other, unnecessary energy consumption occurs and the means of transport become exposed to unnecessary wear.

The ratio of the empty runs is calculated as the sum of the empty runs of the means of transport divided by the sum of the running times of the means of transport.

$$ER_{ratio} = \frac{ER_{total}}{RT_{total}} \tag{4.17}$$

with: ER_{ratio} ratio of empty runs to runtime [s]

ER_{total} total empty runs of the means of transport [s]

RT_{total} total run time of the means of transport [s]

The calculation of the empty runs results from the transport times from the respective sink, for example workstation picking order 1/2, workstation WS 1.1/1.2 or WS 2.1/2.2, workstation wedding/packaging 1/2 or goods issue back to goods receipt.

$$ER_{total} = \sum ER_{PO,m} + \sum ER_{WS,m} + \sum ER_{WP,m} + \sum ER_{GI,m} \tag{4.18}$$

with: ER_{total} total empty runs of the means of transport [s]

$ER_{PO,\,m}$ total transportation time to/from the workstation picking order 1/2 of means of transport m without transport order [s]

$ER_{WS,\,m}$ total transportation time to/from the workstation WS 1.1/1.2 or WS 2.1/2.2 of means of transport m without transport order [s]

$ER_{WP,\,m}$ total transportation time to/from the workstation wedding/packaging of means of transport m without transport order [s]

$ER_{GI,\,m}$ total transportation time to/from the goods issue of means of transport m without transport order [s]

The total time of the empty runs of a means of transport to and from the workstation picking order 1/2 consists of the sum of the transport times to the respective workstation picking order 1/2 without order and from the respective workstation picking order 1/2 to the goods receipt.

$$ER_{PO,m} = \sum_{0}^{q} ER_{PO,x} + \sum_{0}^{q} ER_{PO,y} \tag{4.19}$$

with: $ER_{PO,m}$ total transportation time to/from the workstation picking order 1/2 of means of transport m without transport order [s]

$ER_{PO,x}$ transportation time without order to the workstation picking order x [s]

$ER_{PO,y}$ transportation time from the workstation picking order y to the goods receipt [s]

q number of transport tours

The total time of the empty runs of a means of transport to and from the workstations WS 1.1/1.2 or WS 2.1/2.2 consists of the sum of the transport times to the respective workstation WS 1.1/1.2 or WS 2.1/2.2 without order and from the respective workstation WS 1.1/1.2 or WS 2.1/2.2 to the goods receipt.

$$ER_{WS,m} = \sum_{0}^{q} ER_{WS,x} + \sum_{0}^{q} ER_{WS,y} \tag{4.20}$$

with: $ER_{WS,m}$ total transportation time to / from the workstation WS 1.1/1.2 or WS 2.1/2.2 of means of transport m without transport order[s]

$ER_{WS,x}$ transportation time without order to the workstation x [s]

$ER_{WS,y}$ transportation time from the workstation y to the goods receipt [s]

q number of transport tours

The total time of the empty runs of a means of transport to and from the workstation wedding/packaging 1/2 to the goods receipt consists of the sum of the transport times to the respective workstation wedding/packaging 1/2 without order and from workstation wedding/packaging 1/2 to the goods receipt.

$$ER_{WP,m} = \sum_{0}^{q} ER_{WP,x} + \sum_{0}^{q} ER_{WP,y} \tag{4.21}$$

with: $ER_{WP,m}$ total transportation time to/from the workstation wedding/packaging 1/2 of means of transport m without transport order [s]

$ER_{WP,x}$ transportation time without order to the workstation wedding/packaging x [s]

$ER_{WP,y}$ transportation time from the workstation wedding/packaging y to the goods receipt [s]

q number of transport tours

The total time of the empty runs of a means of transport from the goods issue to the goods receipt consists of the sum of the transport times to the goods issue without order and from goods issue to the goods receipt.

$$ER_{GI,m} = \sum_{0}^{q} ER_{GI,x} + \sum_{0}^{q} ER_{GI,y} \tag{4.22}$$

with:
- $ER_{GI,\,m}$ — total transportation time to/from the goods issue of means of transport m without transport order [s]
- $ER_{GI,\,x}$ — transportation time to the goods issue without order [s]
- $ER_{GI,\,y}$ — transportation time from the goods issue to the goods receipt [s]
- q — number of transport tours

The total time of the runtime of a means of transport is the sum of the total amount of utilisation time and the empty runs.

$$RT_{total} = \sum UT_{total} + \sum ER_{total} \tag{4.23}$$

with:
- RT_{total} — total runtime of the means of transport [s]
- UT_{total} — total utilisation time of the means of transport [s]
- ER_{total} — total empty runs of the means of transport [s]

Chapter 5
Description of the system

This chapter commences with an introduction of basic terms in system theory. With the help of system theory as a proven tool for the presentation of complex systems, a description of the system with its subsystems and system elements, derived from the real production environment, the Werk150, as the subject of investigation of the present work follows. Thereby, the focus is on the description of the relationships between the system elements. Finally, the system of the Werk150 is shown in Figure 5.2.

5.1 System theory

In general, system theory describes the theory of the relationship between the elements of a system, between the structure and function of systems, and between subsystems and overall systems (Ropohl, 2009, pp. 77–78). Systems theory is now widely used in different scientific disciplines. In the field of engineering, systems theory has proven effective to describe complex structures, such as production and logistics systems, using abstract models (Bossel, 2004, pp. 32–33). In the following section, the term system theory is explained according to (Gutenschwager, Rabe, Spieckermann, & Wenzel, 2017, p. 11):

A system is fundamentally limited in its scope and defined by so-called system boundaries with regard to the environment (system environment). Using defined interfaces, a system can exchange matter, energy and information (input and output variables) at the system boundaries. The exchange from the environment into the system is described with input variables, otherwise with output variables. Considering systems in production and logistics, the system interfaces and thus the influences of the system environment on the system are in many cases of high relevance. A system consists of system elements which can represent subsystems or can be considered as not further analysable. Thus, a machine can, in themselves, be part of a production system or considered as a system with its machine components being itself. The structure of a system results from the relationships between the elements of a system. These relationships may be determined by different influences of the system environment or system functionality. Each system element has properties that are mapped using constant and variable attributes, also called state variables. The respective states of a system element are described by the values of constant and variable attributes at a time. The states of the system elements at

a specific time in turn define the system state. The states of the elements may be subject to changes of one or more state variables due to a running process. In this context, process is defined as the sum of all interacting processes in a system that transforms, transports or stores matter, energy, or information. Processes can interact in a sequential, parallel, concurrent and synchronised manner. The individual elements contain their own sequential structure, which is characterised by specific rules regarding state variables and state transitions. The basic terms of system theory are shown in the following figure.

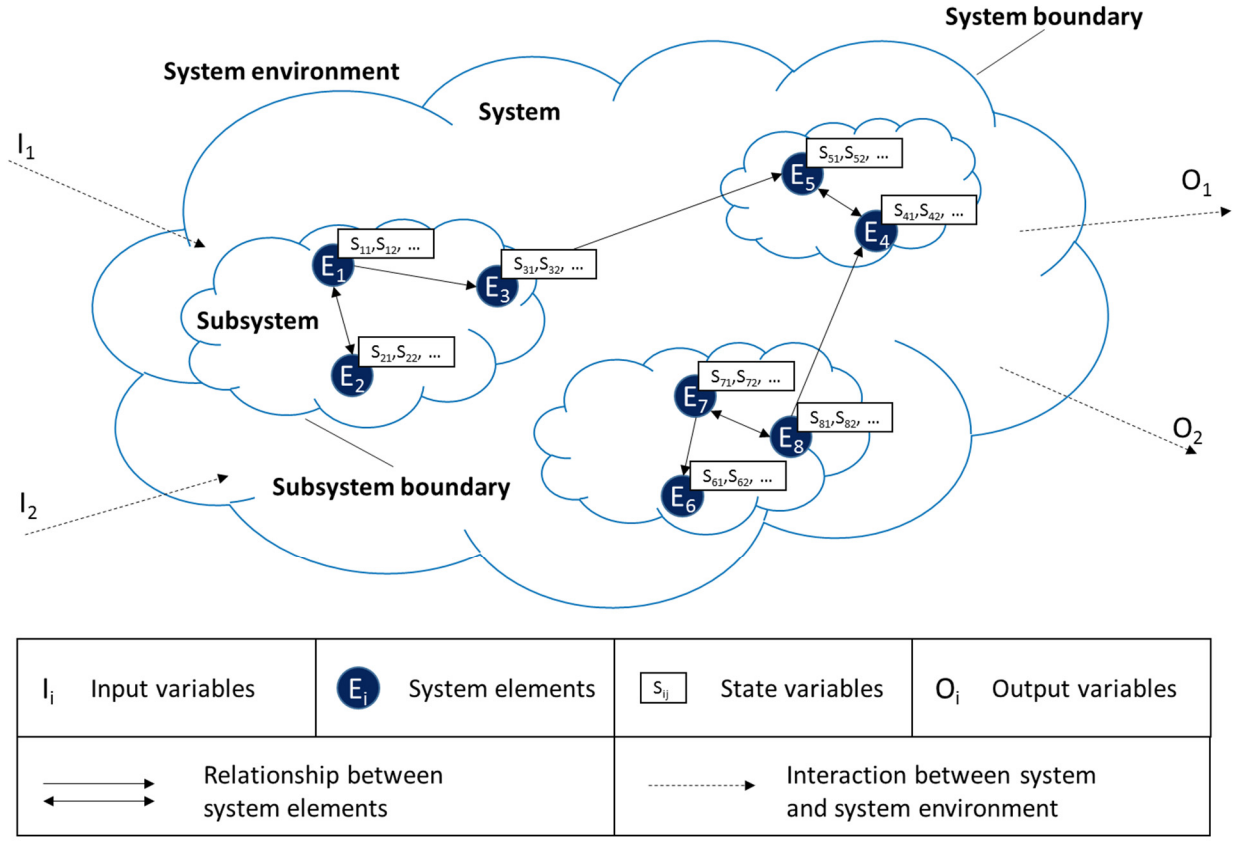


Figure 5.1: Basic terms of system theory based on Gutenschwager (2017, p. 12) and Wiendahl (2010, p. 9)

In general, a distinction can be made between static and dynamic systems. According to Wunsch & Schreiber (2006, p. 73), the values of the output variables for static systems are generated at the same time as the values of the input variables. In system theory, dynamics is defined as the behaviour over time, which is determined by the system states and the state changes, hence the change of input and output per time unit. The behaviour of the environment is referred to as external dynamics, the behaviour of the system elements and subsystems as internal dynamics (Flechtner, 1984, pp. 287–290). In the following section, the considered system, the Werk150, with its subsystems and system elements is presented. Thereby, the system of the Werk150 can be assigned in principle to dynamic systems.

5.2 System of the Werk150

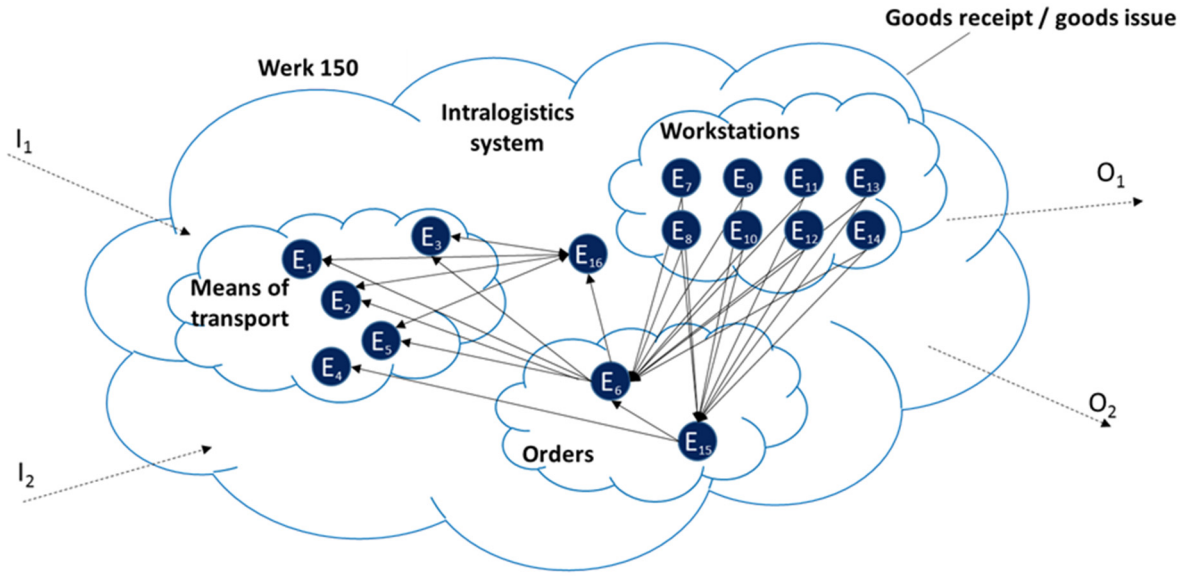
Within the considered intralogistics system of the Werk150, goods receipt and goods issue indicate a system boundary. The system is delimited from its system environment by processes before the goods receipt and after the goods issue. Furthermore, the system is divided into the subsystems ‘means of transport’, ‘orders’ and ‘workstations’. The subsystem ‘means of transport’ includes the various means of transport like AGVs, NeoKu, KollRo and human. The system elements ‘transport order’ and ‘A/C parts’ are referred to as the subsystem ‘order’. The workstation picking order 1/2, the workstation WS 1.1/1.2 and WS 2.1/2.2 and the workstation wedding/packaging 1/2 are composed in the subsystem ‘workstations’. The system element ‘manager’ is not assigned to a subsystem. The system element ‘manager’ represents a virtual agent. The ‘manager’ is responsible for the allocation of the customer orders, an order of a city scooter of the model ‘FlexBlue’ or ‘FlexAir’, and A parts to the means of transport.

The input variables of the system are the generated customer orders for the city scooters of the models ‘FlexBlue’ and ‘FlexAir’. The matching output variables are the tested, error-free and assembled city scooters in the goods issue, which leave the system.

The generated customer order for the production of a city scooter contains several transport orders referred to the system element ‘transport order’. The system element ‘transport order’ checks the queue in front of each workstation in order to determine the shortest waiting time. Then, the system element ‘transport order’ sends its information to the system element ‘manager’. The system element ‘manager’ negotiates with the different system elements of the subsystem ‘means of transport’ except those with the system element ‘KollRo’ and selects a suitable means of transport. In order to execute the transport order, the chosen means of transport receives the required information from the system element ‘transport order’.

The system element ‘transport order’ receives the required processing time at the required workstation from the respective system element of that workstation. At each workstation, the customer order is checked for defective parts. In the case of a defective part, the system element ‘A/C parts’ informs the system element ‘transport order’. The system element ‘transport order’ creates an express order and sends it to the system element ‘manager’. In addition, each workstation checks whether there are enough C parts for the production of a city scooter. If this is not the case, the system element ‘A/C parts’ creates an order and sends it to the system element ‘KollRo’.

The state variables of the system elements are described later in the course of the work. Figure 5.2 shows the considered intralogistics system of the Werk150.



Symbol	Description	Symbol	Description
I_1	Customer orders of the city scooter 'FlexBlue'	E_7	Workstation picking order 1
I_2	Customer orders of the city scooter 'FlexAir'	E_8	Workstation picking order 2
O_1	Finished city scooter 'FlexBlue'	E_9	Workstation WS 1.1
O_2	Finished city scooter 'FlexAir'	E_{10}	Workstation WS 1.2
E_1	AGV 1	E_{11}	Workstation WS 2.1
E_2	AGV 2	E_{12}	Workstation WS 2.2
E_3	Kukaliwa	E_{13}	Workstation wedding/packaging 1
E_4	KollRo	E_{14}	Workstation wedding/packaging 2
E_5	Human	E_{15}	A/C parts
E_6	Transport order	E_{16}	Manager

Figure 5.2: System of the Werk150 with its considered subsystems and system elements

Chapter 6
Development of the algorithm for sequence plan optimisation

This chapter explains the theoretical basics of process modelling. Thus, modelling languages and diagrams which are used in this research study are presented. In addition to the explanation of the terms, a selection is made of the prospective modelling framework representation. Based on the knowledge of modelling, a class diagram based on the system of the Werk150 described in Section 5.2 is first created and explained. The class diagram serves as basis for the process model of the algorithm for sequence plan optimisation. Thereby, the process model is illustrated by an activity diagram. Subsequently, the operations for generating a new sequence plan for the order allocation to the means of transport are presented. The representation of the optimisation algorithm is based on the representation of a job shop scheduling problem by Niehues (2016b) and derives it with respect to the control of intralogistics means of transport. Appendix C shows sections of the source code of the presented operations.

6.1 Modelling basics

When analysing a real or planned system with regard to its functionality or performance, the required analysis of the system itself can only be made to a limited extent. For investigation, models are usually created as images of a system (Gutenschwager et al., 2017, p. 13). The model concept, according to Stachowiak (1973, pp. 131–133) still corresponds to the model concepts used in engineering today, which assumes that a model cannot completely map the examined system, but always represents a simplified image of the system against the background of the model's purpose. Details that are not relevant for the investigation of a system are omitted in the development of the associated model.

Both systems and models can be described by specific characteristics. Some important characteristics in the context of this thesis for classifying systems and models are briefly explained below and shown in Figure 6.1. A distinction can generally be made between physical and intangible models. In particular, verbal and graphical descriptive models are often used to describe the considered system and to develop formal models. Due to their structuring effect, graphics are the ideal engineering instrument for dealing with the discrepancy between

logical-analytical development on the one hand and the holistic recognition of a system on the other hand (Patzak, 1982, p. 129). During the analysis of a system, knowledge can be gained either by analytical calculation or by simulation. Thus, models can be distinguished, according to the type of the used examination method, in analytical models and simulation models. Analytical models are based on systems of equations which reflect the existing system relationships. In contrast to this, in simulation models the model state is updated step by step, whereby simulation models are appropriate for the illustration of the system behaviour (Page, 1991, pp. 4–5). The following figure illustrates the different model classifications.

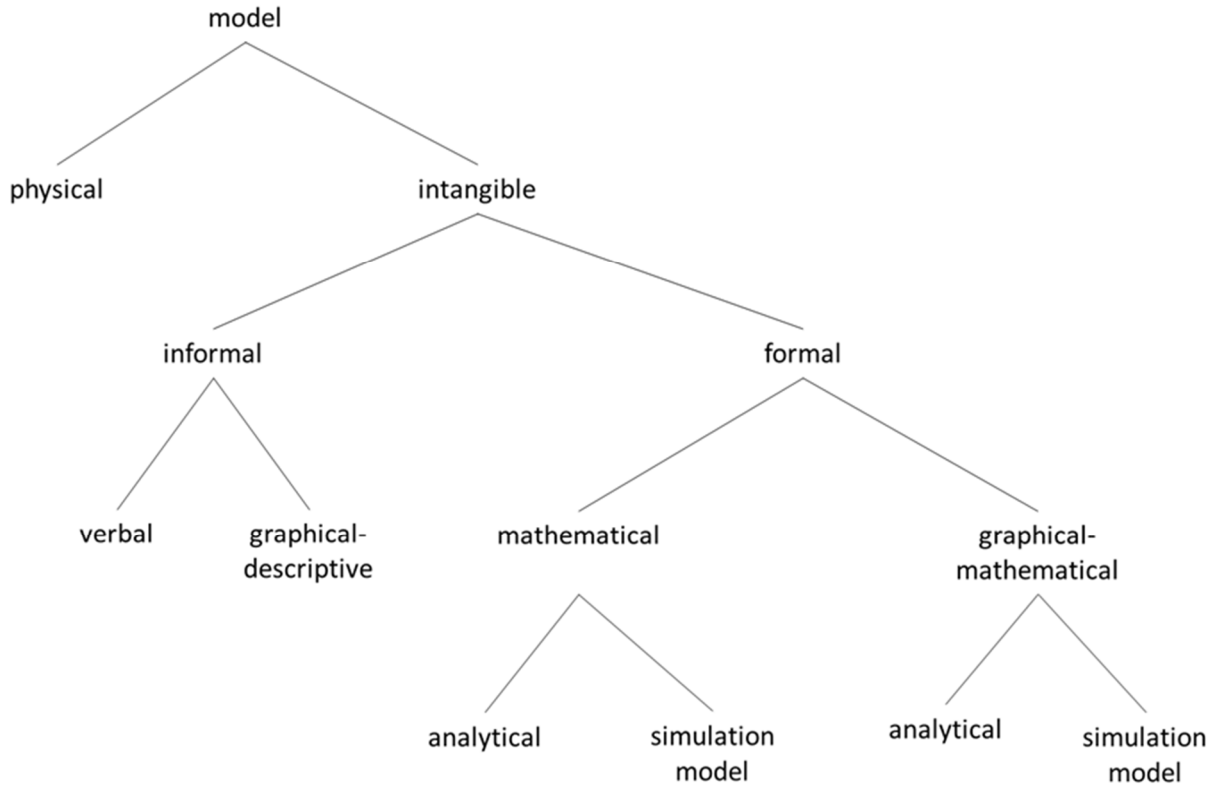


Figure 6.1: Model classification (Page, 1991, p. 5)

Furthermore, models can still be classified in terms of the type of their transition state. In general, static and dynamic models can be distinguished. Dynamic models are subdivided into continuous and discrete models. While the state variables in continuous models can be described by continuous functions, in discrete models the values of the state variables change abruptly at certain times. Both continuous and discrete models can be deterministic and stochastic. For deterministic models, the reaction to a particular input is clearly defined, whereas in stochastic models the responses can only be represented by probability distributions (Gutenschwager et al., 2017, pp. 16–18).

The intended purpose of a model is determined by the research question and the objectives underlying the model study. Models can be subdivided into description models, explanatory models, design models and forecasting models according to their purpose (Trier, Bobrik, Neumann, & Wyssussek, 2013, pp. 57–58).

Descriptive models have a documentary character and are thus often used to support the system demarcation. For this reason, they are also referred to as investigation models. Typical description models are organisational charts for describing organisational structures and process models for mapping operational processes. In cases where not all relevant characteristics of a considered system can be determined within the scope of a system survey, explanatory models are used. An explanatory model based on theories depicts the structure of the original system in such detail that it can be used to explain the observed system behaviour. Design models are used for designing systems and selecting measures in project planning and serve as a decision support. The goal of a forecasting model is the prediction of future system states under given assumptions (Trier et al., 2013, pp. 57–58).

In the context of this research study, the developed model is classified as an intangible, formal, graphical-mathematical simulation model. The research study aims to model the control of intralogistics means of transport in logistical sequence planning. Thereby, the model is based on the material and information flows of the intralogistics system. For this purpose, a dynamic model based on discrete events is developed. Therefore, the description of the model takes place in form of an explanatory model.

6.2 Model language and diagrams

The choice of a model for the representation of a system is determined decisively by the respective application, modelling purpose and the chosen perspective. The model language and diagrams that are important to the author and are used in the course of the work are described in the Unified Modeling Language (UML).

The UML is a standardised, formal language for the specification, visualisation, construction and documentation of object-oriented systems. It provides various diagrams that allow the modelling object to be viewed from different angles, like structural diagrams, such as the class diagram for representing static structures and behavioural diagrams for example, the use case diagram and the activity diagram for representing dynamic structures (Bobrik & Trier, 2013, p. 87). The diagrams used in this paper are the class diagram and the activity diagram.

Class diagrams: The class diagram consists of classes, attributes, and relationships. Real or abstract things are referred to as a class, which are of interest for the considered section of the operational task. A class can only be represented by its name, or additionally with its attributes and methods. Individual classes can be linked by relationships. The simplest form is a connection line with an arrow indicating the reading direction of the connection. The numbers at the end of lines are called multiplicities and determine the ratio of objects of one class related to the objects of the other class. In an aggregation, an unfilled diamond is at one end of the line. This means that the class at the end of the line without diamond is part of the class with diamond. An even stronger connection is marked by a solid diamond. In this case the class cannot exist without the whole (Bobrik & Trier, 2013, pp. 87–88).

Activity diagram: Activity diagrams are a combination of flowcharts and Petri nets. They represent activities and their temporal and logical sequence in sequential, parallel and alternative paths. In activity diagrams, control flows, data flows and object flows can be mapped. Each activity diagram contains at least one initial state and one end state. The individual actions are presented by rectangles with rounded corners, which are connected to a path by a control flow. Control flows are used to show the transition from one activity state to the next one. A decision node is used to make a decision. A swimlane notation groups related activities into one column, similar to creating a function in a program. The outgoing arrows from the decision node can be labelled with conditions.

Alternative paths are generated by conditional branches, also called forks. Horizontal bars split a path into multiple, parallel paths, or merge parallel paths into one path. In this way, concurrency can be modelled. If paths are merged by a join notation, the process stops at this point until the flows of all paths have arrived. Paths can also be united by a simple merge, in this case, there is no synchronisation (Bobrik & Trier, 2013, pp. 89–90).

6.3 Class diagram of the Werk150

The autonomous system for intralogistics control described requires high data availability. Figure 6.2 shows a class diagram with its classes, attributes and relationships. A unique identification number identifies each object.

At the core of this class diagram is the manager who receives information about transport orders and allocates them to the means of transport. The manager selects the individual means of transport according to certain target criteria in order to ensure a high target fulfilment. Therefore, the manager uses different methods to search the solution space in order to generate

an optimal or near-optimal solution (+generatePopulation(), +calcFitness(), +mutatePopulation(), +crossoverPopulation(), +selectPopulation()). For the calculation, the manager needs the information about the transportation times, transportation start, and utilisation times of the means of transport. Each means of transport is thus able to calculate their transportation start (+calcTransportationStart()) and time (+calcTransportationTime()) according to their position, speed and availability. Thus, the means of transport are able to calculate their utilisation time (+calcUtilisationTime()) and as well as the amount of time regarding empty runs (+calcEmptyRuns()). Based on this information, the manager decides about the order allocation according to the best possible target fulfilment. Then, the manager 'negotiates' with the means of transport, which can themselves modify the solution based on local knowledge (+mutatePopulation()). Finally, the best-found solution so far will be taken for the allocation of the order to the means of transport.

For the calculation of the optimal solution, the transport order sends the required information to the manager. The information consists of the city scooter model, the current position and its next workstation. After the transport order is allocated to a specific means of transport, the transport order sends the required information to the respective means of transport. In order to select its next workstation, the transport order receives the waiting times in front of the possible workstations (+getWaitingTime()). The transport order receives the information about the processing time from the respective workstation (+getProcessingTimes()). In addition, the transport order calculates its target figures 'short order waiting time and transportation time' (+calcTransportationTime()) and 'high adherence to schedules' (+calcScheduleDeviation()).

At each workstation the city scooter is checked for possible damage or scratches on the components. If a component of the city scooter is damaged, a new transport order is created and then sent to the manager. In addition, for each city scooter the availability of the required C parts is checked. The information on the C parts is stored in the components class. The principle for reordering C parts is based on a pull system. If a certain stock of C parts is used up, the KollRo receives the information to restock the production line.

As soon as the KollRo receives an order, an optimal route for the transport of the C parts is calculated. Therefore, the KollRo uses the same methods (+calcFitness(), +mutatePopulation(), +crossoverPopulation(), +selectPopulation()) as the manager in order to find an optimal or near-optimal solution. For the calculation, the KollRo requires information from the C parts about its destinations and desired delivery time. Then, the manager calculates the transportation times (+calcTransportTime()), transportation starts (+calcTransportationStart()) for the

sequence plan. Additionally, the KollRo can check the stock of C parts on its waggons (+checkTrainLoad()). Thus, the KollRo knows exactly when to return to the supermarket to refill its waggon with new C parts. Furthermore, the KollRo is able to calculate its logistics target figures 'short order waiting and transportation time' (+calcTransportationTime()), 'high adherence to schedules' (+calcScheduleDeviation()), 'short utilisation time' (+calcUtilisationTime()) and 'minimum ratio of empty runs to runtime' (+calcEmptyRuns ()). Figure 6.2 shows a class diagram of the Werk150 with its classes, attributes and relationships.

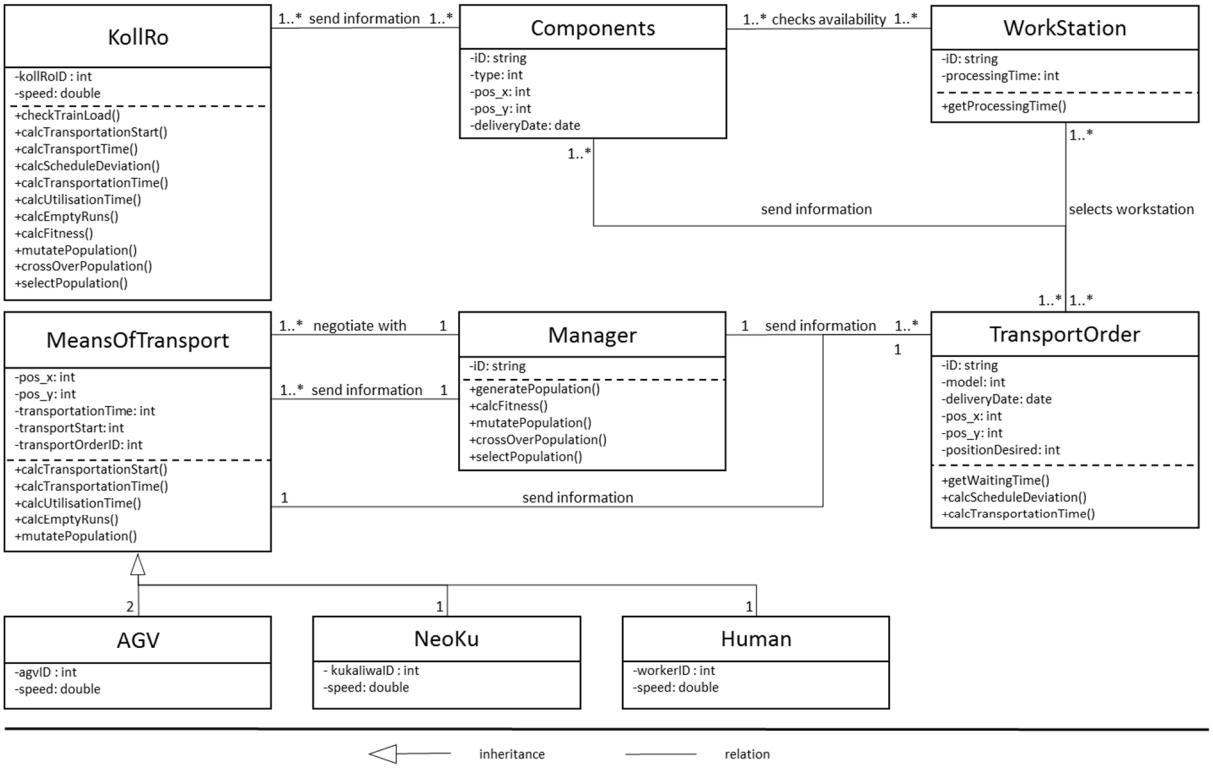


Figure 6.2: Class diagram of the Werk150

6.4 Process of the algorithm for sequence plan optimisation

For optimal use of the means of transport according to the defined logistics target figures, the solution space is searched with the help of an algorithm. The aim is to find the best possible solution in a reasonable time. According to the logistics target criteria mentioned in Section 4.2, the algorithm considers several objective functions. The formulas for calculating the objective functions are the equivalent of those already described, logistics target criteria. The objective functions distinguish between the target system of the means of transport AGV, Human and NeoKu, which are responsible for the transport of the city scooters, and the target system of the KollRo, which supplies the production line with C parts.

This means that the following objective functions apply for the target system of the means of

transport. The objective function minimisation of 'schedule deviation of order i' (SD_i) is equivalent to the target figures 'high adherence to schedule' and minimisation of 'transport time of order i' (TT_i) as a function of the target figure 'short order waiting time and transportation time'. Minimisation of the 'ratio of empty runs to utilisation time of all means of transport' (ER_m) is used as an objective function for the logistics target figure 'minimum ratio of empty runs to utilisation time'. Subsequently, the aim of a uniform distribution of the means of transport is expressed be the objective function 'utilisation time of the means of transport m' (UT_m) according to the target figure 'balanced distribution of the utilisation time'.

$$\min SD_i \tag{6.1}$$

$$\min TT_i \tag{6.2}$$

$$\min ER_m \tag{6.3}$$

$$UT_m = 0.25 \tag{6.4}$$

Regarding the objective functions of the KollRo, the objective function minimisation of 'schedule deviation of order c' (SD_c) is equivalent to the target figure 'high adherence to schedule'. The objective function to minimise the 'transport time of order c' (TT_c) as a function of the target figure 'short order waiting time and transportation time' and the minimisation of the 'ratio of empty runs to utilisation time of the KollRo' (ER_{KR}) is used as an objective function for the logistics target figure 'minimum ratio of empty runs to utilisation time'. Subsequently, the minimisation of the 'utilisation time of the KollRo' (UT_{KR}) is considered according to the target figure 'short utilisation time'.

$$\min SD_c \tag{6.5}$$

$$\min TT_c \tag{6.6}$$

$$\min ER_{KR} \tag{6.7}$$

$$\min UT_{KR} \tag{6.8}$$

Referring to the review of the state-of-the-art literature, the applied approach is based on a genetic algorithm with neighbourhood search to iteratively search the solution space. The process of the algorithm is shown in Figure 6.3, consisting in its core of a population comprising a certain number of chromosomes. Each chromosome corresponds to the solution in the form of the representation rule presented in the following section. During the solution search new solutions are generated from the chromosomes by using different operators. If these solutions show an improvement, the fitness evaluation and the selection of the solution for the

new population of the following generation is executed. After the optimisation run is determined, the best solution is applied for the intralogistics control.

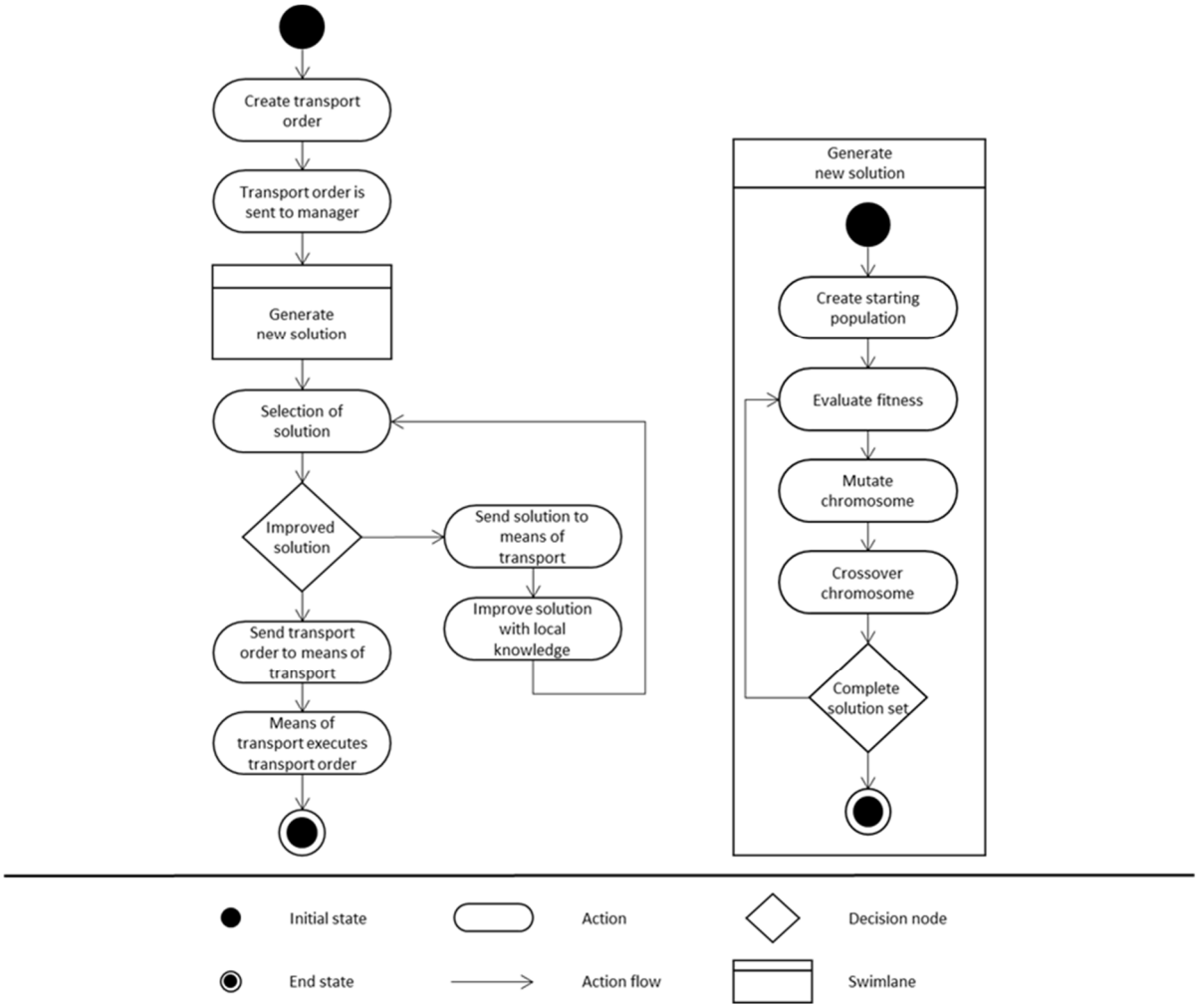


Figure 6.3: Activity diagram of the algorithm for sequence optimisation based on Niehues (2016b, p. 114)

6.4.1 Representation of the sequence plan

The use of operators to create new sequences requires a representation of the sequence plan in a form, with respect to mutation and crossover operators, that allows a simple revision of the initial solution. The transport order sequence represents the sequence of the means of transport as well as the transport orders in the form of a matrix. This enables the separate calculation of values depending on the sequence plan, such as adherence to delivery times and values, depending on the means of transport, such as transport times.

Within the period under review, the transport order sequence consists of a number of customer orders O_1 to O_i and a set of means of transport mot_1 to mot_m. In the following discussion, an order is defined as a physical product, in this case the city scooter ordered by the customer. Each order consists of several transport orders. The transport orders refer to the sequence of transporting the order from a specific source to the desired sink. Therefore, each order i consists of a number n_j of transport orders $O_{i,j}$ with $j \in \{1, ..., n_j\}$. Each transport order $O_{i,j}$ is assigned to a start time of the transport $TS_{i,j}$. With respect to the sequence plan, each transport start $TS_{i,j}$ of $O_{i,j}$ can be assigned a position $p_{i,j}$ in the global order, which is the starting point for the representation of the transport order.

The matrix referred to below as the order sequence matrix (OS) is structured in such a way that a transport order can be exactly assigned to one means of transport. For each chromosome of the population an OS_c matrix with $c \in \{1, ..., n_c\}$ is created. The OS_c matrix contains the global order sequence in the form of the positions of all start times of the transports:

$$OS_c = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,n} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{m,1} & p_{m,2} & \ldots & p_{m,n} \end{pmatrix} \tag{6.9}$$

The creation of the OS_c matrix from a sequence plan is described in the following section.

6.4.2 Generating an initial population

An initial solution is created by determining the position numbers $p_{i,j}$ as a function of the start of transport $TS_{i,j}$ of the transport orders. The value of $p_{i,j}$ is randomly inserted in the field with column n and line m in the OS_c matrix without violating the technical restrictions of the means of transport. If two or more transport operations have simultaneous transport starts, $TS_{i,j}$, consecutive values for $p_{i,j}$ must be assigned to the corresponding transport operations since each number can only exist once within the OS_c matrix. It is not relevant which value is assigned to which transport order. If a means of transport is not required for the transport of an

order, the corresponding field within the matrix is assigned the value 0. Figure 6.4 shows an example of transferring a sequence plan into an OS_c matrix.

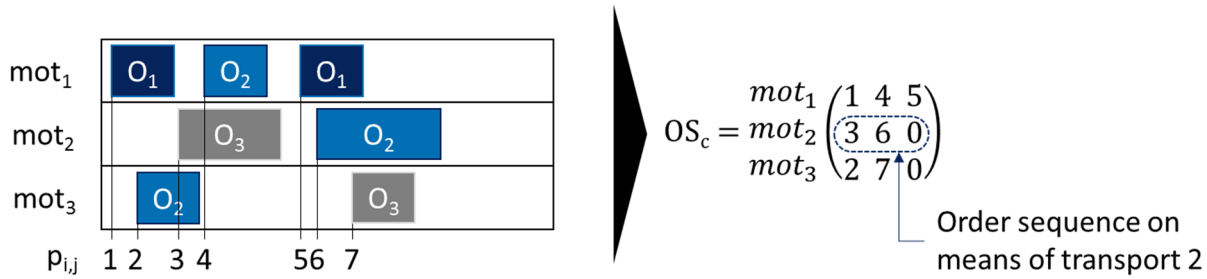


Figure 6.4: Exemplary transfer of a sequence plan into the OS_c matrix

The coding of the sequence plan as OS_c matrix is carried out every time the algorithm is started since the numbering of $p_{i,j}$ is always based on the currently valid sequence plan. This sequencing process is described in the following section.

6.4.3 Calculation of the fitness function

Good solutions are solutions that contain high adherence to schedules, short transportation times and a balanced utilisation time of the means of transport. Such solutions should be assigned a high fitness value in order to increase their likelihood of being included in the population of the next generation. The calculation of the fitness value takes place for each solution s_c of each chromosome $c \in \{1, ..., n\}$ of its generation. After the sequencing of the associated OS_c matrix, the solution value s of the $OS_{c,s}$ matrix is set in relation to the solution value of the existing chromosomes $OS_{c\text{-}1,s}$. If the solution value s of $OS_{c,s}$ outperforms an existing solution, the new solution is accepted and the existing solution is removed.

$$OS_{c-1,s} \leq OS_{c,s} \tag{6.10}$$

with: $OS_{c\text{-}1,s}$ solution value of the OS matrix of the existing chromosomes [s]

$OS_{c,s}$ solution value of the OS matrix of the new chromosome [s]

To decode the OS_c matrix of a valid solution, the positions $p_{i,j}$ are checked in ascending order and the times of transport start $TS_{i,j}$ and end $TE_{i,j}$ of the associated transport are calculated. The following times are relevant for the calculation of $TS_{i,j}$:

- Actual simulation runtime t
- Earliest availability of means of transport mot_m, corresponds to the transport end of the previous transport $TE_{i,j-1}$ on means of transport mot_m

If $TE_{i,j-1}$ is smaller than the simulation runtime t, t is set as $TS_{i,j}$. The time at the end of transportation $TE_{i,j}$ depends on the transportation time $t_{m,i,j}$ and is calculated as follows:

$$TE_{i,j} = \begin{cases} t + t_{m,i,j} \\ TS_{i,j} + t_{m,i,j} \end{cases} \tag{6.11}$$

with: $TE_{i,j}$ transport end of transport order j of customer order i [s]

$TS_{i,j}$ transport start of transport order j of customer order i [s]

$t_{m,i,j}$ transportation time of the means of transport m for transport order j of customer order i [s]

t simulation run time [s]

First, the value of the solution of each generation is calculated as the sum of the schedule deviation, transport time and subsequently the number of empty runs of the means of transport.

$$s_c = SD_i + TT_i + ER_{ratio} \tag{6.12}$$

with: s_c numerical solution value of chromosome c [s]

SD_i schedule deviation of the order i [s]

TT_i transport time of the order i [s]

ER_m ratio of empty runs to utilisation time of all means of transport m [s]

After the calculation of the value of the solution for each chromosome, the solution is sent to the means of transport mot_m with the lowest utilisation time. The means of transport is able to propose a new solution s_m. If the new solution has a higher fitness value than the old solution, the new solution with its sequence of the means of transport is executed, otherwise the old solution is maintained.

$$s_c \leq s_m \tag{6.13}$$

with: s_c numerical solution value of chromosome c [s]

s_m proposed solution value of the means of transport m [s]

6.4.4 Selection operator

The selection operator determines which generation is randomly chosen for the operator's mutation and crossover to generate solutions for the next generation of the new population. The selection operator is usually based on probabilities. The probability $p_{c,s}$ of selecting a chromosome depends on the fitness of its solution s compared to the sum of all fitness values of the solution set S. The solution set S is composed of the sum of all the solutions obtained. Thus, a solution with a high fitness value has a higher chance of being selected than a solution with a lower fitness value. Solutions with lower fitness values should not be excluded for the operators in principle, since by mutating or crossover of solutions with a lower fitness value, the outcome can still result in a higher fitness value of the solution. In addition, considering solutions with a lower fitness value also leads to an increase of a larger solution space and a higher possibility to escape local optima.

Since the algorithm aims to minimise the objective functions, the numerical value of the fitness value should be as low as possible. Therefore, the fitness values must first be converted before applying the selection operator. The following formula is therefore used to determine the probability $p_{c,s}$ for solution s of each chromosome c.

$$p_{c,s} = 1 - \frac{s_c}{\sum_0^c s_c} \tag{6.14}$$

with: $p_{c,s}$ probability of the chromosome c to be selected based on its solution s [%]

s_c numerical solution value of chromosome c [s]

Each generation represents a share of a bar according to their probability, where 0 equal to 0 % is the bottom of the bar and 1 equal to 100 % the top. The cumulated probabilities of the generations add up to 1. After the determination of the probability according to each solution, a random number $r \in \{0, ..., 1\}$ is generated. Depending on the range of corresponding probabilities in which the value of the randomly generated number is, its generation is selected.

Figure 6.5 shows an example of randomly selecting a chromosome based on its solution. The value of the complete solution set S is 500. In this example there are three different chromosomes with different solution values. The first chromosome has a solution value of 50, which corresponds to an acceptance probability of 60 %. The second chromosome has a solution value of 150 with the corresponding acceptance probability of 30 %. Lastly, the third chromosome has a solution value of 300 and thus an acceptance probability of 10 %.

In the first scenario, the randomly generated number r has the value 0.43. This value is within the probability range of the first chromosome, which is thus selected. In the second scenario,

the value of the random number r is 0.74 and therefore within the probability range of the second chromosome. In this case, chromosome two is selected.

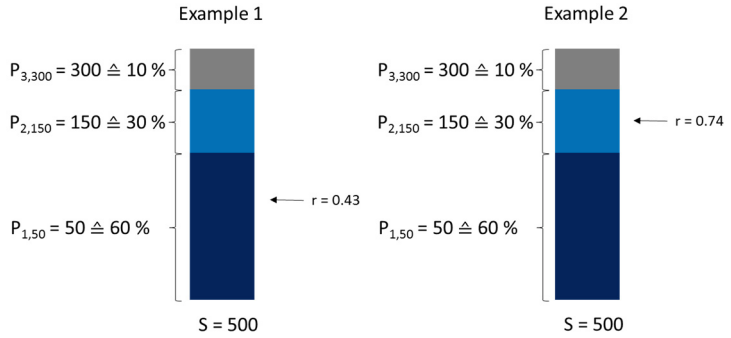


Figure 6.5: Exemplary presentation of randomly selecting a generation of a population

6.4.5 Sequencing

An important part of neighbourhood search procedures is the design of the operators, which seek new solutions in the neighbourhood, also referred to as ‘sequencing’. Regarding the optimisation of the sequence, a solution should not differ too much from the original solution in order to avoid unnecessary planning insensitivity in intralogistics.

In terms of short computation time, the sequencing is designed in such a way that worse solutions are avoided. In principle, it can be assumed that the sequencing was optimal or at least sufficiently good before a new event occurs, e.g. new incoming order or disturbances.

6.4.5.1 Mutation operator

A new solution is created by swapping two values of $p_{i,j}$ in the OS_c matrix. By swapping the two values, the order sequence for the means of transport is changed. As a result of these changes, a new sequence is created that differs from the initial solution. The swapping of two elements is called a mutation according to the GA. In order to avoid a violation against technological restrictions of the different means of transport, the swapping of the two values always takes place in the same line. Thereby, the means of transport as well as the two chosen values are selected randomly.

In the example shown in Figure 6.6, the means of transport mot_1 was selected randomly. The values at the positions $p_{1,1}$ and $p_{1,3}$ are swapped. By swapping the two values, a new solution is generated.

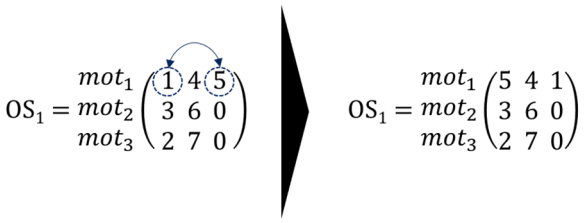


Figure 6.6: Example of the mutation operator of the values $p_{1,1}$ and $p_{1,3}$

6.4.5.2 Crossover operator

In contrast to mutations, crossover involves the simultaneous exchange of several elements. This can lead to a greater modification of the solution. During the crossover operation, the lines of two OS_c matrices are swapped. This requires an additional repair procedure to detect duplicated values and correct them, since each value of the OS_c matrix is only allowed to occur once. Since the allocation of the orders to the means of transport results from the technological restrictions, the OS_c matrix is only valid if the order sequence of each line corresponds to the technological restrictions of the respective means of transport. For this reason, even though the chosen lines are randomly selected, only the same lines of different OS_c matrices can be swapped. This ensures that the new solution cannot be invalid from a technological point of view. This simplifies the repair function for the crossover operator.

In the following paragraphs, the crossover operator with repair function is described by using two examples (see Figure 6.7). In both examples the first lines of each of the two OS_c matrices are swapped. Thus, the values in the first line of the second OS_2 matrix are overwritten into the first OS_1 matrix.

Regarding Example 1, the swapping of the two lines of the two matrices OS_1 and OS_2 leads to

an invalid solution, because the transport order 5 at the position $p_{1,1}$ has been lost in the new matrix OS_{1*}. In order to convert the new, but still invalid solution, into a valid solution, it must be repaired with the following step. The repair function checks the new matrix OS_{1*} for completeness. In this example transport order 5, which was at position $p_{1,1}$ in the original matrix OS_1, has been lost; it will be added to its original line in the new matrix OS_{1*} at the last position. After the execution of the repair function the new matrix OS_{1**} has a valid solution.

By swapping the first lines of the two matrices OS_1 and OS_2 in Example 2, transport order 6 now occurs twice at the positions $p_{1,1}$ and $p_{2,2}$ in the new matrix OS_{1*}. In this case, the repair function works as follows. The first step is to delete transport order 6 at position $p_{2,2}$ in the new matrix OS_{1*}. A duplicate order is never deleted in the newly added line, so that a larger deviation from the original solution is generated. At the same time, transport order 1 moves one column to the left, where transport order 6 was placed before. After the execution of the repair functions the new matrix OS_{1**} now has a valid solution.

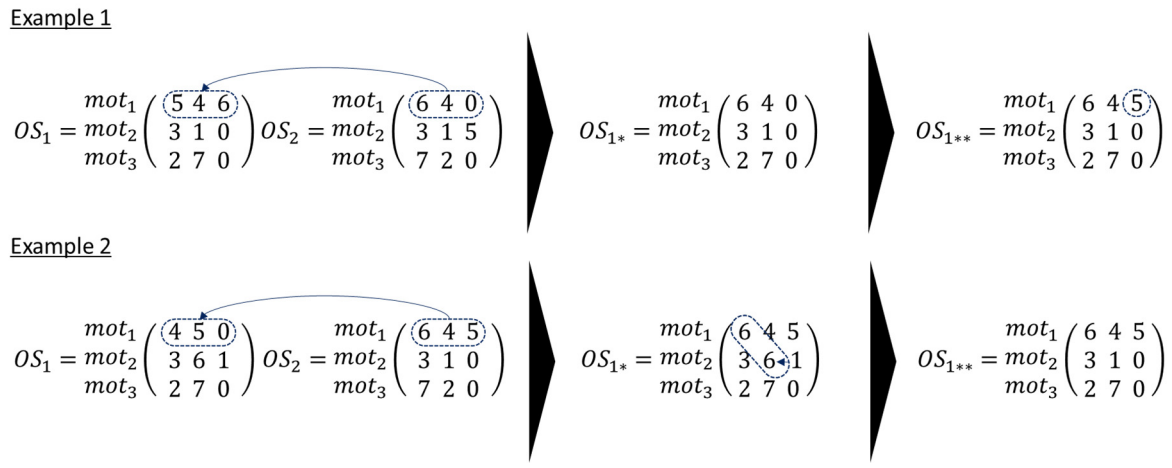


Figure 6.7: Examples of the crossover operator of the first line with repair function

6.4.5.3 Mutation operator based on local knowledge

With the last operator, the best transport order sequence so far, the chromosome with the best solution value is sent to the means of transport with the lowest utilisation time. The aim is to improve the quality of the solution and to balance the utilisation times of the means of transport. In this process, the selected means of transport randomly chooses a transport order from another means of transport. The transport order is then checked against possible technological violations. If the technological restrictions are not violated, the transport order is added to the order sequence of the means of transport and in turn removed from the order sequence of the other means of transport. However, if the means of transport cannot fulfil the technological restrictions for the selected transport order, the transport order is not transferred.

In Figure 6.8, for example means of transport mot_1 has the lowest utilisation time of all means of transport. The means of transport randomly selects the transport order at position $p_{1.2}$. In this example, the technological restrictions are not violated and therefore transport order 3 is added to the means of transport mot_1. Because transport order 6 was deleted from the order sequence of means of transport mot_2, every transport order which was at the right-hand side of transport order 6, moves one column to the left.

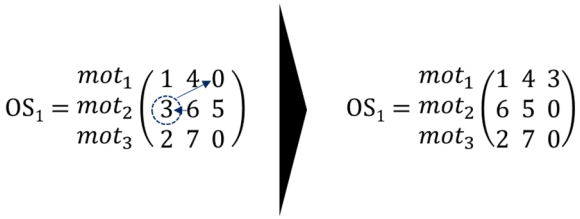


Figure 6.8: Example of the mutation operator based on local knowledge of mot_1 selecting the value $p_{2,2}$

6.4.6 Termination

In order to avoid an infinite loop of the algorithm, a termination criterion is required, which defines the end of an evolutionary algorithm. It is recommended to terminate the program if the overall fitness of the population does not improve over several generations. Due to the requirement of a short response time in the context of adaptive intralogistics control, there is also an upper limit for the calculation time after which the algorithm must be terminated. After the termination of the optimisation run, the best solution is selected and the sequence plan is applied to the intralogistics control of the means of transport.

Chapter 7
Implementation in the simulation

This chapter commences with the description of general terms of simulation. The simulation software AnyLogic was used for the development of the simulation model. The software is introduced in this chapter and the simulation methods are presented that are applied for modelling the previously described algorithm for sequence optimisation of the means of transport. Then, the algorithm is implemented in the simulation model. The different flow charts and the interactions of the modelled agents are described.

7.1 Simulation basics

In the systematic analysis of complex logistics, material flow and production systems, mathematical-analytical methods often reach their limits due to the diverse interdependencies within the considered technical system. In comparison, the standard VDI 3633 states that a simulation is able to examine and evaluate time-related behaviour of complex technical systems over a period of time on the basis of a simplified model of reality (Verein Deutscher Ingenieure, 2018, pp. 3–4). According to VDI 3633 simulation is the "representation of the system with its dynamic processes in an experimental model with the aim of reaching findings which are transferable to reality". (Verein Deutscher Ingenieure, 2018, p. 28)

There are different simulation methods for the simulation of dynamic systems. The standard VDI 3633 differentiates these methods into continuous and discrete simulation concepts regarding simulation time and change of state of the underlying model (Verein Deutscher Ingenieure, 2018, p. 29). In continuous simulation, the model state changes steadily with time, while in discrete simulation, state changes occur abruptly at discrete points in time because of a specific event (Mattern & Mehl, 1989, p. 201). The discrete simulation can in turn be subdivided into time-controlled and event-controlled simulation methods with regard to the control of the simulation process. Fixed or variable steps in the time-controlled simulation control the simulation time. Only the events between the last and the new point in time are executed. Regarding event-driven simulation, the simulation time always continues with the next temporal event, until the event is subsequently executed. Based on the modelling style used, the event-driven simulation can be further subdivided into event-driven, activity-oriented, process-oriented and transaction-oriented simulations (Mehl, 1994, pp. 2–4).

Figure 7.1 shows the classification of simulation methods.

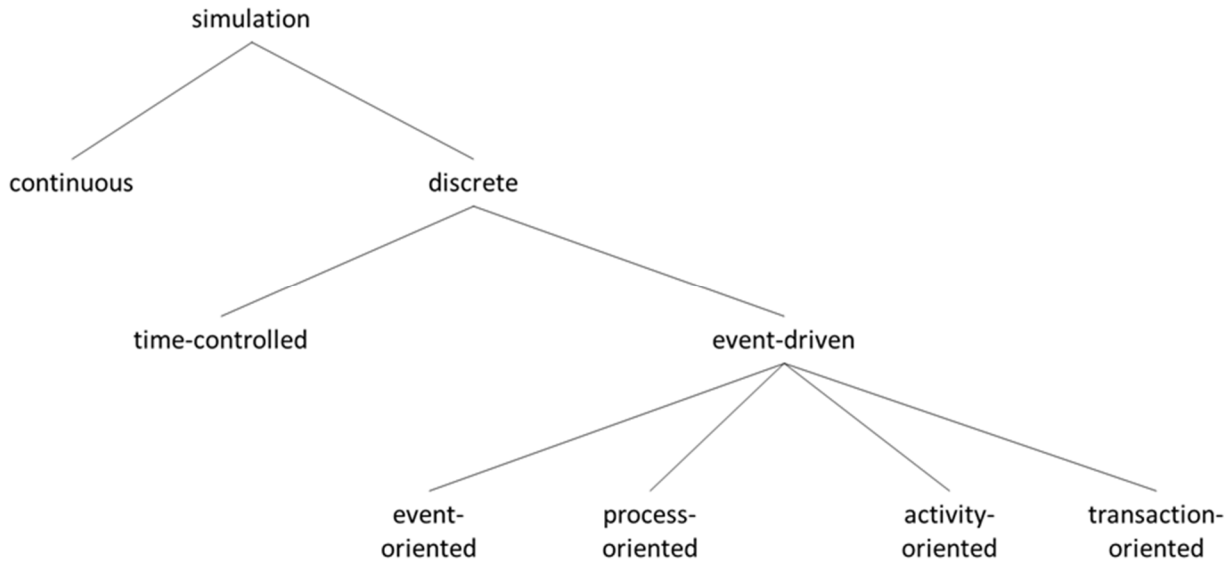


Figure 7.1: Classification of simulation methods(Mehl, 1994, p. 4)

While in event-oriented simulation, changes in reality are displayed as events in the model, in activity-oriented simulation they are regarded as activities of a certain period, modelled by the activity of the beginning and end of an event. Process-oriented simulation is characterised by related state changes of the real world with a modelled process that represents a sequence of related events (Mehl, 1994, pp. 4–5). Transaction-oriented simulation represents a special case of process-oriented simulation and will therefore not be elaborated on further.

The simulation model of this thesis can be classified as a discrete simulation model based on an event-driven approach. In order to simulate the intralogistics system of the Werk150, a process-oriented simulation is applied for simulating the real system.

7.2 Modelling software AnyLogic

For the implementation of the developed system, a suitable simulation software was initially searched for. Therefore, a benefit analysis of simulation software in the area of production and logistics was undertaken. The evaluation of the benefit analysis can be found in Appendix D.

Based on the results of the benefit analysis, the simulation software AnyLogic 8 Personal Learning Edition 8.5.1 was selected to develop the simulation model. AnyLogic provides a single platform in the field of dynamic simulation modelling. The simulation software offers three different simulation methods for development:

- Discrete event modelling
- Agent-based modelling
- System dynamics

These three methods can be used in any combination within the software, to simulate business systems of any complexity. The development platform offers a graphical drag-and-drop environment for the simple creation of models to insert new agents, functions, variables, parameters and other operations. Additionally, the AnyLogic software environment is fully mapped into Java code and linked with the AnyLogic Simulation engine, therefore becoming a completely independent stand-alone Java application. This enables AnyLogic models to be run on any Java-enabled environment or even in a web-browser as applets. Java is an object-orientated programming language with high performance to enable the modeller to define and manipulate data structures of any complexity as well as developing efficient algorithms (Borshchev, 2013, p. 380).

7.2.1 Selection of the simulation methods

Simulation modelling is used in a wide and diverse range of applications. The various applications can be classified by their level of abstraction. The right abstraction level is critical for the success of the modelling project. The abstraction level of a model can be linked with the corresponding simulation method. There are three existing simulation methods, each serving a particular range of abstraction level, namely system dynamics, discrete event modelling, and agent-based modelling. System dynamics is mainly used at a high abstraction level, often for strategic modelling. Discrete event modelling operates at a medium and medium-low abstraction level. The range of agent-based modelling supports a high abstraction level as well as a very detailed view. Figure 7.2 shows the different modelling methods according to their abstraction level (Borshchev, 2013, pp. 35–36).

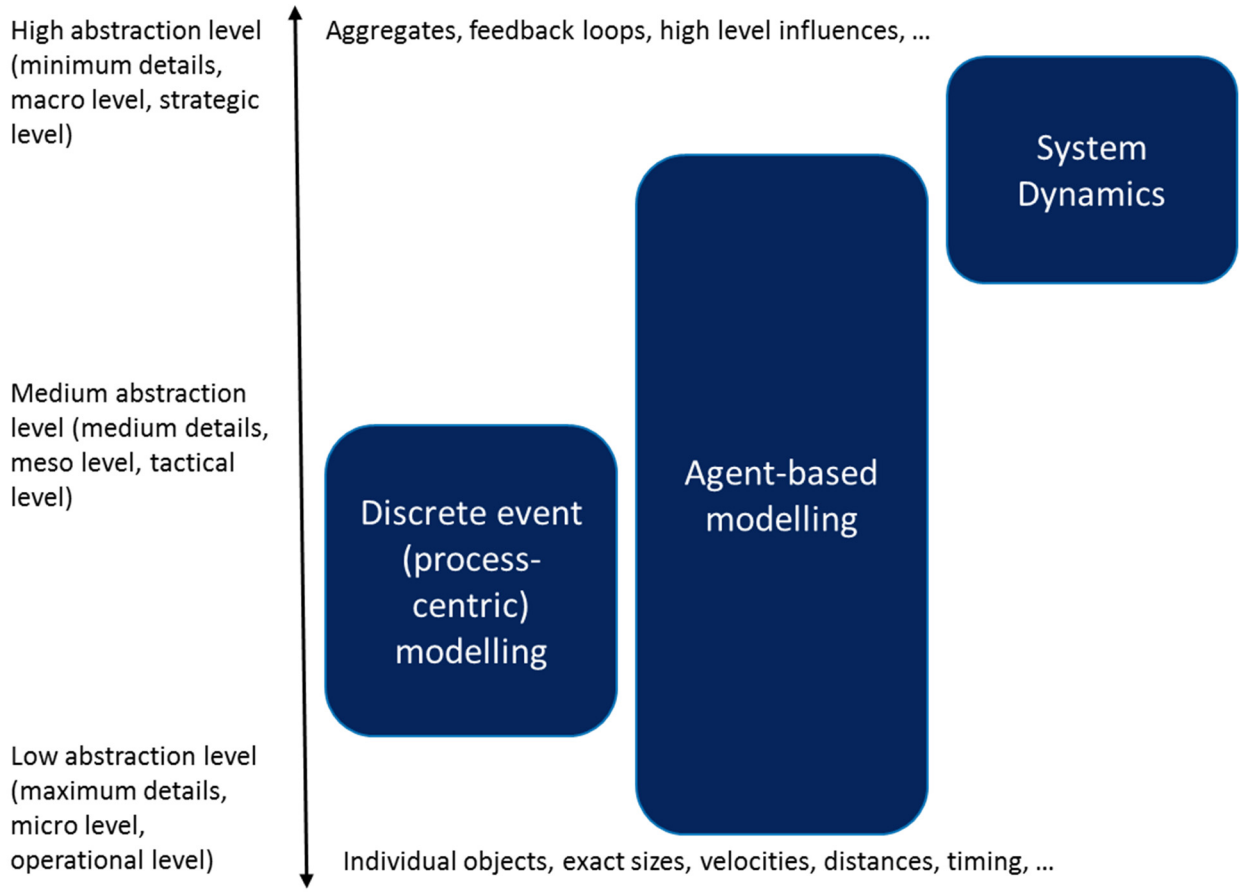


Figure 7.2: Methods in simulation modelling (Borshchev, 2013, p. 36)

For this thesis, a multi-method approach is chosen to model the autonomous system. For the control of intralogistics means of transport, a medium-low abstraction level is chosen. Therefore, discrete event modelling and agent-based modelling are applied for the model. The different workstations are modelled with the discrete event approach because they respond to discrete events. The means of transport, the transport order, the components and the manager are modelled with the agent-based approach focusing on individual objects and describing their local behaviour according to rules. In the following section, the two simulation methods are described in more detail.

7.2.2 Description of the simulation methods

With respect to this thesis, the term 'method' is understood as a general framework for mapping a real world system to its model. The methods which are described in particular in the following section are discrete event modelling and agent-based modelling. The two methods are briefly described and the building blocks used in the work are explained.

7.2.2.1 Discrete event modelling

This modelling paradigm considers real-world processes as an ordered sequence of events, although the system being simulated exists in continuous time. Changes in the simulation model take place at specific events, which occur at a specific point in time (Stewart, 2009, p. 680). The discrete event modelling approach is applied especially for systems that can be described by a sequence of operations. AnyLogic provides the user with a process modelling library for modelling the specific system. With the help of the objects provided in the library, the real-world systems can be modelled in terms of agents (e.g. transactions, customers, products, parts and vehicles), processes (e.g. sequences of operations typically involving queues, delays and resource utilisation) and resources. The building blocks used for this work are described in the following table.

Table 7.1: Overview of the building blocks used in the development of the model

Icon	Name	Description of the building block
	Enter	Agents are inserted into a specific point of the process model
	Exit	Removes the incoming agents from the process flow but does not delete the agent.
	MoveTo	The agent moves to a new location.
	Delay	The agents are delayed for a certain amount of time. The delay time can be evaluated dynamically, stochastically or depends on the agent.
	Queue	In this case a queue equals a buffer. The agents wait to be accepted by the next object in the process flow.
	SelectOutput	The agents are directed to one of the two output ports depending on probabilistic or deterministic condition or on the agent as well as on any external factors.
	RackStore	The agent is put into a cell of a given pallet rack, or RackSystem. The RackStore moves the agent from its current location in the network to the cell location.
	RackPick	The agent is removed from a cell in the specified pallet rack, or RackSystem and moved to the specified destination location.
	Convey	The incoming agents are transported by conveyors to the specified destination point of a conveyor network.

7.2.2.2 Agent-based modelling

From the viewpoint of practical applications, agent-based modelling focuses on individual objects and describes their local behaviour according to local rules. For the agent-based approach agents, such as people, companies, projects, vehicles, cities and products have to be identified and their behaviour, e.g. main drivers, reactions and memory has to be defined. Agents are able to interact with each other as well as with the dynamic environment (Borshchev, 2013, pp. 49–50). Following the bottom-up approach, the global behaviour emerges as a result of the interactions of many concurrent individual behaviours. The behaviour of an agent can be represented by a number of variables, parameters and functions. To model the agent's behaviour, AnyLogic provides its users with a set of agent components. The agent components used for the simulation model in this work are presented in Table 7.2.

Table 7.2: Overview of the agent components used in the development of the model

Icon	Name	Description of the agent component
	Agent	Agents are main building blocks of AnyLogic model. An Agent can have behaviour, memory, timing and contacts.
	Event	An event is used to schedule some action in the model.
	Parameter	A parameter describes characteristics of objects statically. A parameter is usually a constant in a single simulation and is not changed during the simulation.
	Variable	Variables are often used to store the results of model simulation. Variables can change during the simulation runtime.
	Function	Function will return the value of an expression by calling the associated function. Functions can be used multiple times in the model.
	Collection	Collections present a group of objects into a single unit. With collections, it is possible to store, retrieve and manipulate aggregate data.

The agent's behaviour is defined with a statechart consisting of states and transitions. Each state describes the behaviour of the agent by certain actions, events or a combination of functions, variables or parameters (Borshchev, 2013, p. 287). Exiting transitions define the reaction of a state. Each transition may be triggered by user-defined conditions, such as timeouts or rates, messages received by the statechart and Boolean conditions. Table 7.3 explains the different elements of a statechart and the various types of triggers to activate a transition.

Table 7.3: Overview of the elements of a statechart used in the development of the model

Icon	Name	Description of the elements of a statechart
	Statechart Entry Point	Statechart entry point is the initial state of a statechart. Each statechart consists of exactly one statechart entry point.
	Final State	At the final state, its action is executed and the statechart terminates.
	State	A state reacts to conditions and/or events. In the event of a taken transition, the statechart switches from one state to another.
	Timeout	The transition is triggered after the specified amount of time, which may be stochastic or deterministic.
	Rate	Rate triggered transitions are similar to timeout triggered transition, however the rate is a form of updatable exponential timeout, thus has a stochastic recurrence time.
	Condition	If a certain Boolean condition becomes true, the transition is triggered. The associated action is executed, when this condition becomes true.
	Message	A transition occurs when the statechart receives a message from another agent.
	Agent arrival	The transition is triggered when an agent arrives at its destination. The agent's movement was initiated by calling agent's moveTo() function.
	Branch	By using branches it is possible to create a transition that has more than one destination state as well as several transitions that merge together.

7.3 Structure and control of the simulation

This section describes the structure and control of the simulation model. The simulation model consists of the Main agent, the Manager agent, the TransportOrder agent, the AGV agent, the NeoKu agent, the Human agent, the KollRo agent and the Component agent. The programmed agents with their statecharts are presented. Special attention is paid to the processes and the communication of the agents.

7.3.1 Main agent

The most important agent is the Main agent, which serves as the primary modelling environment in which all other agents reside. The agents Manager, TransportOrder, AGV, Human, NeoKu, KollRo and Components were created in the Main agent. The individual agents are described in more detail in the following sections. Furthermore, the production environment was created in the Main agent, based on the production environment of the Werk150 including goods receipt, goods issue, workstation order picking 1/2, workstation WS

1.1/1.2 and WS 2.1/2.2 and workstation wedding/packaging 1/2. The considered logistic target figures in this work are visualised in the form of tables in the Main agent. In addition, in the Main agent the customer orders for the production of a new city scooter are simulated. The creation of the orders is controlled by an event. Every transport order for the order of a city scooter is sent via message to the TransportOrder agent to be further processed. The occurrence of the event has a stochastic recurrence time. The specific attributes for the respective transport order such as product type and actual and desired position are described by variables. The workstation order picking 1/2, workstation WS 1.1/1.2 and WS 2.1/1.2 and workstation wedding/packaging 1/2, which the city scooter passes through during its path towards completion in production, are programmed in the Main agent. The individual workstations and their process steps are explained below.

7.3.1.1 Workstation order picking 1/2

The procedure at the workstations order picking 1/2 for the stem and base is in principle the same. The process starts at the ‘Enter 1’ block. Here, the required A parts are checked for any damage. If an A part is damaged, a transport order is sent to the Manager agent. The order of the city scooter cannot be processed until the needed A part has been delivered. Therefore, the customer order moves to the ‘Delay 1’ block. If the A parts are not damaged, the customer order moves directly to the ‘MoveTo 1’ block. After the arrival of the correct A part, the process is continued and the A parts of the city scooter are placed on a fixture. This process takes place in the ‘MoveTo1’ block. Then, the customer order waits in the queue in front of the workstation and waits to be processed (‘Queue 2’ block). At the ‘Delay 2’ block, the TransportOrder agent transmits the corresponding processing time to the workstation. As soon as the individual A part has been placed on the fixture, it is transported to the defined pick-up location by a conveyor belt (‘Convey 1 block’). The customer order moves to the ‘RackStore 1’ block. At this block, the customer order is placed on a shelf and a new transport order is sent to the Manager agent. The order remains on the shelf (‘Delay 3’ block) until a means of transport picks it up (‘RackPick 1’) and transports it to the next workstation.

At the ‘SelectOutput 2’ block, the destination of the next workstation is transmitted to the customer order. The base moves to either workstation WS 1.1 (‘MoveTo 2’ block) or WS 2.1 (‘MoveTo 3’ block), the stem moves either to workstation WS 2.1 (‘MoveTo 2’ block) or WS 2.2 (‘MoveTo 3’ block). The process flow at the workstation order picking 1/2 is shown in Figure 7.3.

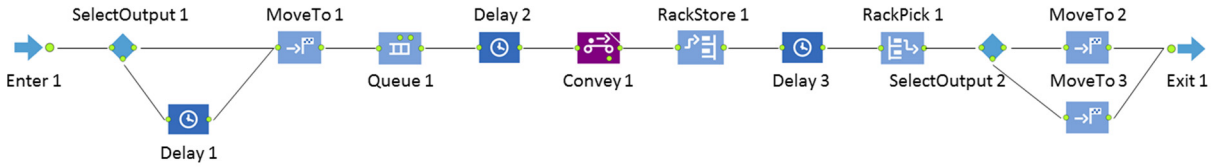


Figure 7.3: Process flow at the workstation order picking 1/2

7.3.1.2 Workstation WS 1.1/1.2 and WS 2.1/2.2

The process flow at the workstations WS 1.1, WS 1.2, WS 2.1 and WS 2.2 do not differ from each other. At the ‘Enter 2’ block, the incoming order is first checked for any damage at the workstation. If a component is damaged, a transport order is sent to the Manager agent. The order cannot be further processed until the correct A part has been delivered. The customer order moves to the ‘Delay 4’ block and waits for the A part. As soon as the correct A part is delivered, the assembly process is continued. In case of there being no damaged A parts, the customer order moves directly to the ‘SelectOutput 4’ block. The supply of the required C parts at the workstations is provided by a pull system. Therefore, the first step is to check whether the minimum stock of an individual C part has been reached. This process happens at the ‘SelectOutput 4’ block. If the minimum stock level has been reached for a C part it is reordered. A transport order is sent to the KollRo agent for this purpose. The KollRo agent receives the corresponding information regarding location, quantity, and delivery time from the Component agent. If a C part is missing for processing the city scooter, the process is stopped at this point until the required C part is delivered. The customer order moves to the ‘Delay 5’ block. After the arrival of the missing C part the process can be continued. The ‘Delay 5’ block is skipped if all required C parts are available. Then, the customer order waits in the queue in front of the workstation and waits to be processed (‘Queue 2’ block). The corresponding processing time is transmitted to the workstation by the TransportOrder agent. The duration time in the ‘Delay 6’ block corresponds to the processing time at the workstation. As soon as the assembly process step is finished, the order is placed on a shelf and a new transport order is sent to the Manager agent. The order remains on the shelf until a means of transport picks it up and transports it to

the next workstation wedding/packaging 1/2. The process flow at the workstations WS 1.1, WS 1.2, WS 2.1 and WS 2.2 is presented in Figure 7.4.

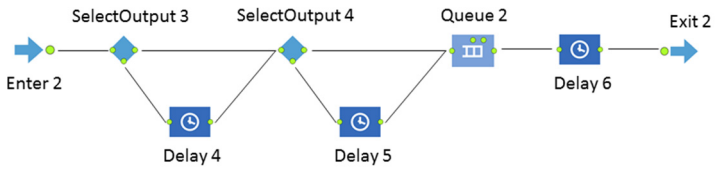


Figure 7.4: Process flow at the workstations WS 1.1, WS 1.2, WS 2.1 and WS 2.2

7.3.1.3 Workstation wedding/packaging 1/2

The procedure at the workstation wedding/packaging 1/2 is similar to the above-mentioned procedure of the previous workstations. First of all, the incoming order is checked for any damage and in the case of damage, a transport order is sent to the Manager agent. This process takes place at the 'Enter 3' block. The order is delayed until the correct A part has been delivered ('Delay 7' block). As soon as the correct A part is delivered, the assembly process is continued. The customer order moves to the 'SelectOutput 5' block. The customer order moves directly to the 'SelectOutput 5' block if the customer order is not damaged. At the 'SelectOutput 5' block, the customer order is also checked if the minimum stock of the required C part for this processing step is reached. Analogous to the workstations WS 1.1/1.2 or WS 2.1/2.2, if the minimum stock level has been reached for a C part the required C part is reordered. A new transport order is sent to the KollRo agent. The KollRo agent receives the corresponding information regarding location, quantity, and delivery time from the Component agent. If a required part for the assembly of a scooter is missing, the process stops until the required C part is delivered ('Delay 8' block). The process can be continued after the arrival of the missing C part. Then, at the 'MoveTo 4' block, the customer order checks if its counterpart has already arrived at the workstation. As long as the corresponding counterpart has not yet arrived at the workstation, the customer order waits in the 'Delay 9' block. As soon as the counterpart arrives the corresponding processing time is transmitted to the workstation by the TransportOrder agent. The customer order moves to the 'Delay 10' block and waits for the duration of its respective processing time.

As soon as the assembly process step is finished, it is placed on a shelf and a new transport order is sent to the Manager agent (‘RackStore 2’ block). The order remains on the shelf until a means of transport picks it up and transports it to the goods receipt. The process flow at the workstation wedding/packaging 1/2 is shown in Figure 7.5.

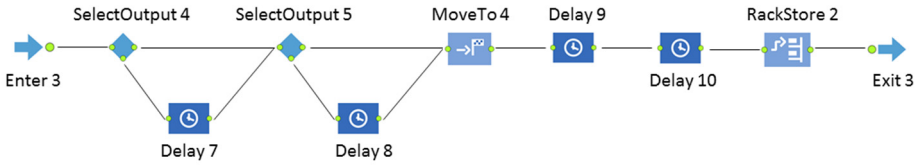


Figure 7.5: Process flow at the workstation wedding/packaging 1/2

7.3.2 Manager agent

The Manager agent has the task of optimally allocating the means of transport according to the logistical target criteria described in Section 4.2 to transport the various orders. Therefore, the approach of a genetic algorithm with neighbourhood search, according to Section 6.4, is applied. The manager agent distributes the transport orders among the means of transport AGV 1, AGV 2, NeoKu and human. Since the Manager agent only controls customer orders and individual transport orders for A parts, the KollRo is not controlled by the Manager agent. The processes of the Manager agent are now explained in more detail.

The Manager agent waits in the ‘waitForOrder’ state until it receives a transport order from the Main agent or a means of transport. The creation of a new transport order can have different reasons such as a new customer order, the failure of a means of transport, or a damaged A part.

As soon as a new transport order arrives, the Manager agent is triggered by a ‘message’ and transits to the next state. In the ‘initializePopulation’ state, all existing transport orders are randomly allocated to the means of transport, taking into account their technical restrictions. This is done for all existing chromosomes in the population, thus creating a valid initial population based on the actual transport orders. After generating the initial population the ‘condition’ trigger is set to true and the Manager agent moves to the ‘selectChromosome’ state.

In the ‘selectChromosome’ state, the individual chromosomes and their solution value are calculated. To determine the solution value, the Manager agent asks the individual means of transport for their current position, their earliest possible transport start, and their speed. From the TransportOrder agent, the Manager agent receives information about its source and sink as well as its desired delivery time. For the additional steps the solution values of the individual chromosomes are normalised. Thus, the solutions can be expressed as a percentage and the

degree of the solution quality can be shown in comparison to the other solution values. After the selection of the chromosome based on its fitness value, the ‘condition’ trigger is set to true and the Manager agent moves to the ‘mutateChromosome’ state.

The selected chromosome is mutated in the ‘mutateChromosome’ state. The mutation process is analogous to the process described in Section 6.4.5.1. After the mutation operator is executed, the worst existing solution before the event of the mutation is selected, deleted and replaced by the new, mutated solution. This process ensures that new solutions are always added to the population and that the searched solution space is enlarged. The new population is then re-evaluated and mutated until the termination criterion is reached. As soon as the termination criterion is met, the Manger agent passes the ‘branch’ transition and moves to the next state.

The crossover operator follows the mutation of the individual chromosomes and their search for a better solution. This process takes place in the ‘crossoverChromosome’ state. This is a further step to enlarge the solution space in order to find new solutions and at best exceeds the solutions found so far. Similar to the mutation operator, the crossover operator probabilistically selects the two chromosomes based on the fitness of the solution value. The crossover operator is carried out according to the process described in Section 6.4.5.2. Since the simple swapping of the transport orders of the two chromosomes can lead to duplications or to a loss of the transport orders, the repair function is subsequently applied. The exact procedure of the repair function is also described in Section 6.4.5.2. According to the principle of the mutation operator, the worst existing solution is selected, deleted and replaced by the new solution. Then the new generation is evaluated again, and the crossover operation is carried out until the termination criterion is reached. As soon as the termination criterion is met, the Manger agent passes the ‘branch’ transition and moves to the state ‘sendOrderMOT’.

In this state, the best solution found so far is selected and sent to the means of transport with the shortest utilisation time. The means of transport can modify the solution based on local knowledge and send the new, modified solution back to Manager agent. The aim of this process is to achieve a balanced distribution of utilisation times of the means of transport. The process based on the mutation of the solution with local knowledge was described in Section 6.4.5.3. The means of transport sends its modified solution back to the Manager agent. Thereby, the Manager agent is triggered by a ‘message’ and moves to the ‘proposalMOT’ state.

In the last state, the Manager agent compares the best solution found with the modified solution of the means of transport. If the modified solution has a higher solution value, this solution is accepted, otherwise the best solution found so far is selected by the Manager agent. The solution with the best fitness value regarding the order allocation is sent to the different means of transport. The ‘condition’ trigger is set to true and the Manager agent returns to its initial state ‘waitForOrder’ and waits for a new transport order. The statechart of the Manager agent is presented in Figure 7.6.

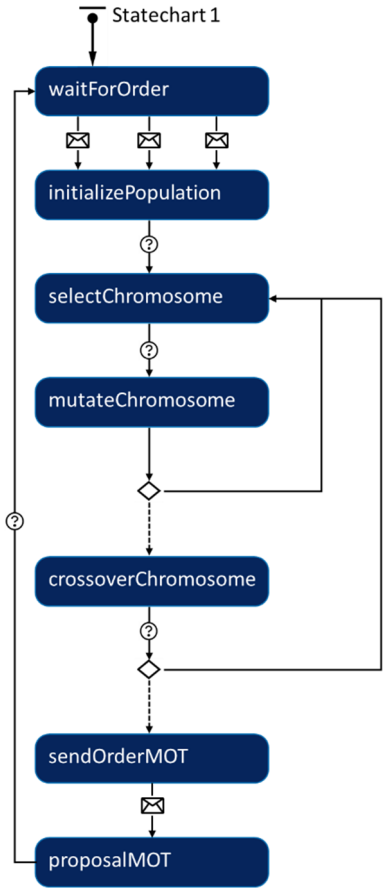


Figure 7.6: Statechart of the Manger agent

7.3.3 TransportOrder agent

After the simulation start, the TransportOrder agent resides in the ‘waitForOrder' state. As soon as an order of a new city scooter has been created in the Main agent, its corresponding transport order is sent to the TransportOrder agent. The TransportOrder agent receives the required information about the transport order from the Main agent, such as product type, source, sink, and delivery time. Thereby, the TransportOrder agent is triggered by a ‘message’ and transits to the state ‘orderPicking’. At the ‘orderPicking’ state, the TransportOrder agent divides the order into two separate orders for the stem and the base. This separation is necessary since the stem and the base are processed at different workstations with different processing times. The two orders are treated separately until they are later reunited at the workstation wedding/packing 1/2. The TransportOrder agent checks the possible sinks for the transport orders. Therefore, the workstations located at the Main agent send information about the queue of orders in front of the workstation to the TransportOrder agent. The agent selects the next sink depending on the shortest waiting time, hence shortest queue length. After the selection of the workstation, the order is transported to the respective workstation order picking 1/2. The ‘condition’ trigger is set to true and the TransportOrder agent moves to the ‘delayTime’ state.

At the ‘delayTime’ state, the TransportOrder agent sends the required information for processing the order to the workstation. The TransportOrder agent stays in this state until the ‘message’ “orderPickingFinished” is sent back to the agent. Then the TransportOrder agent moves to the next state.

After being processed at the workstation order picking 1/2 the order waits to be picked up by a means of transport and is delivered to the next workstation WS 1.1/1.2 or WS 2.1/2.2. The TransportOrder agent waits in the state ‘waitForMOT’ until the agent is triggered by a ‘message’. As soon as the ‘message’ triggers the agent, a means of transport picks up the transport order and is moving it to its next destination. The TransportOrder agent enters the state ‘movedToWorkstation’. Once the means of transport has unloaded the transport order at the desired destination, the TransportOrder agent is triggered by a ‘message’ and transits to the state ‘delayTime2’.

At the ‘delayTime2’ state, the workstation WS 1.1/1.2 or WS 2.1/2.2 receives the required processing information from the TransportOrder agent. As soon as the order is processed, the TransportOrder agent is triggered by a ‘message’ and transits to next state.

At the ‘waitingForMOT2’ state, the TransportOrder checks whether its counterpart has already moved to the workstation wedding/packaging 1/2 or not. If the counterpart has not yet arrived at the workstation wedding/packaging 1/2, the TransportOrder agent moves to the state ‘waitForWedding’. Thereby, the TransportOrder agent is triggered by the ‘message’ “waitForWedding”. While the TransportOrder agent waits in its state, the transport order is picked up by a means of transport and brought to the workstation wedding/packaging 1/2. The TransportOrder agent stays in this state until a ‘message’ of its counterpart arrives. As soon as the this ‘message’ arrives, the processing of stem and base have been performed, hence a new city scooter is produced and moved to the goods issue. The TransportOrder agent transits to the ‘movedToShipment’ state. After two seconds the TransportOrder moves to the ‘final state’ and deletes itself.

If the counterpart has already arrived at the workstation wedding/packaging 1/2, the TransportOrder agent moves to the state ‘waitForMOT’. Thereby, the TransportOrder agent is triggered by the ‘message’ “MOTFound”.

After processing at the workstation WS 1.1/1.2 or WS 2.1/2.2, the order waits to be picked up by a means of transport and is delivered to the next workstation wedding/packaging 1/2. The TransportOrder agent waits in the state ‘waitForMOT’ until the agent is triggered by a ‘message’. The next steps are analogous to the process described above until the TransportOrder agent transits the state ‘waitingForMOT’ again. At this point in time, the stem and the base were both already processed, hence a new city scooter was produced. The finished city scooter is picked up by a means of transport and moved to the goods issue. As soon as the means of transport arrives at the goods issue, the TransportOrder agent is triggered by a ‘message’ and moves to the ‘movedToShipment’ state, before it finally transits to the ‘final state’ and deletes itself. The city scooter is ready for delivery to the customer. The process steps of the TransportOrder agent described above are shown in Figure 7.7.

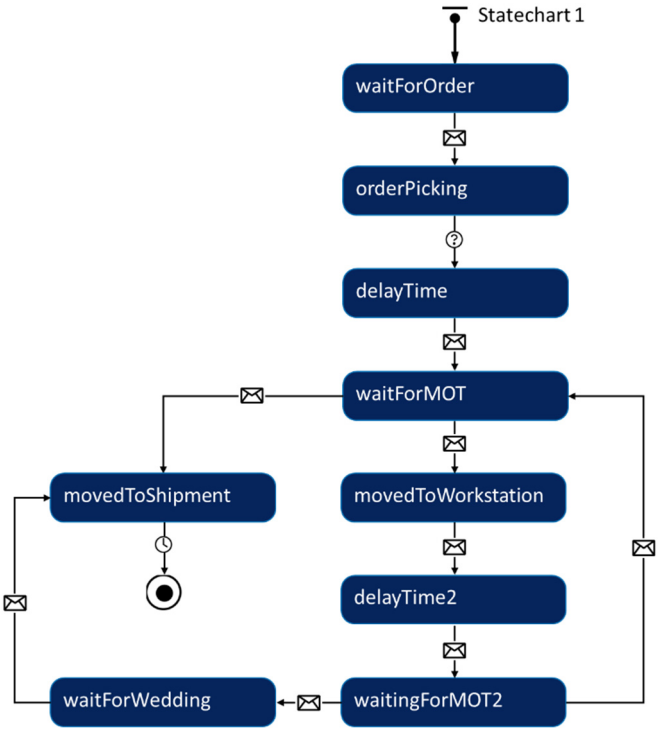


Figure 7.7: Statechart of the TransportOrder agent

7.3.4 AGV agent, NeoKu agent, Human agent

In principle, the process flow for the agents of the means of transport AGV, NeoKu and human, shown in Figure 7.8, is the same. The agent of the means of transport waits in the 'moveToHomeOfMOT' state. As soon as the Manager agent has completed the allocation of the transport orders, a transport order is sent to the selected means of transport. The agent is triggered by a message and transits to the 'branch' transition. At the 'branch' transition the agent checks for available transport orders. If the agent has a transport order to fulfil, the required information about source and sink are stored and the agent moves to the 'pickUpOrder' state. If there is no transport order to be transported, the agent transits back to the 'moveToHomeOfMOT' state and waits for the next transport order. At the 'pickUpOrder' state the means of transport locates its own position in the production environment and calculates its route according to the location of the transport order. After the localisation process, the means of transport moves to the pick-up location of the transport order. Once the means of transport arrives at the pick-up location, the agent is triggered by an 'agent arrival' transition and the agent transits to the 'loading' state. The agent resides at the 'loading' state for a certain time to simulate the loading of the order on the means of transport. Then the 'time out' transition gets activated and the agent moves to the next state.

As long as the agent resides in the ‘dropOffOrder’ state, the means of transport moves the transport order to its desired destination. During the transportation process, the means of transport might fail. The event of a failure is programmed by a ‘rate’ transition. As soon as the ‘rate’ transition is active, the means of transport has a failure and stops moving. The agent transits to the ‘failure’ state. At the ‘failure’ state, the means of transport locates its position and sends the information about its transport order with the updated x- and y-position of the means of transport to the Manager agent. The Manager agent saves this information of the transport order so that the next means of transport that receives the transport order knows where to pick the order up. If a means of transport fails, it loses its privilege to continue executing the transport order. The means of transport loses its transport order and the Manager agent calculates a new sequence plan to allocate the transport orders based on the current information. After sending the information to the Manager agent, a ‘time out’ is triggered and the agent of the means of transport transits to the ‘delay’ state. The agent stays in the ‘delay’ state until the means of transport is repaired. Then, the ‘timeout’ transition is activated and the agent transits to the ‘branch’ transition. At the ‘branch’ transition the agent checks again for available transport orders. If the agent has a transport order to fulfil, the required information about source and sink are stored and the agent moves to the ‘pickUpOrder’ state. If there is no transport order to be transported, the agent transits back to the ‘moveToHomeOfMOT’ state and waits for the next transport order.

If no failure of the means of transport occurs during the transport, the agent is triggered by the ‘agent arrival’ transition, once the means of transport arrives at the delivery station. The agent moves to the ‘unloading’ state and resides at the ‘unloading’ state for a certain time to simulate the unloading process. Then, the ‘time out’ transition gets activated and the agent moves to the next state.

The ‘bringBackFixture’ state can be triggered by an ‘agent arrival’ or a ‘condition’ transition. The ‘agent arrival’ transition is activated, if the means of transport transports a finished city scooter to the goods issue. In this case the means of transport transports the fixture of the delivered order back to the workstation order picking 1/2 before it is able to execute the next transport order. The agent moves to the ‘branch’ transition and checks for available transport orders. If the means of transport did not move a finished scooter to the goods issue, the ‘condition’ transition is activated. The agent moves directly to the ‘branch’ transition and checks for new transport orders. If the agent has a transport order to fulfil, the required information about source and sink is stored and the agent moves to the ‘pickUpOrder’ state. If

there is no transport order to be transported, the agent transits back to the 'moveToHomeOfMOT' state and waits for the next transport order.

In addition to the execution of the transports, the agents of the means of transport can also influence the allocation of the transport orders. The 'proposeSolution' state is triggered by a 'message' sent from the Manager agent. Thereby, the means of transport with the lowest utilisation time is given the opportunity to change the allocation of the transport orders. The means of transport has the goal of achieving a higher utilisation time. Hence, by mutating the existing solution of the Manager agent, the means of transport generates a new solution. The exact operation of optimising the order sequence plan has already been described in Section 6.4.5.3. The new solution proposal of the means of transport is sent back to the Manager agent, who in turn finalises the allocation of the transport orders to the means of transport. After mutating the order sequence sent by the Manager agent, the agent waits in the 'proposeSolution' state for the next 'message'.

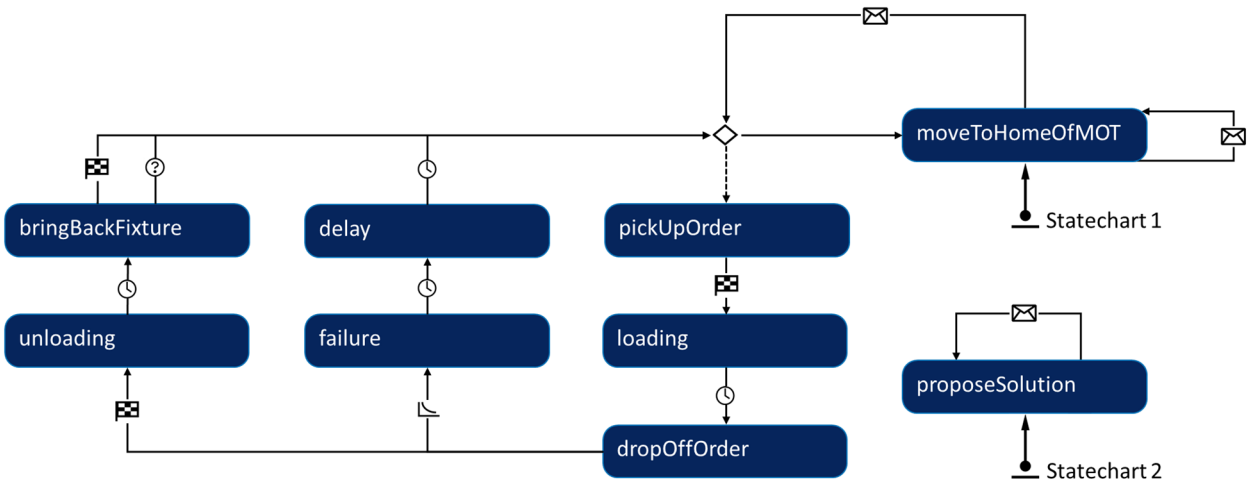


Figure 7.8: Statechart of the Agv/NeoKu/Human agent

7.3.5 KollRo agent

The KollRo must be considered separately from the means of transport, since the KollRo is the only means of transport responsible for the supply of the production line with C parts. The process flow of the KollRo agent is different compared with the other means of transport. The KollRo agent has, on the one hand, a process flow for the transport of the C parts and on the other hand, a process flow for the optimisation of its transport tour. The process flows of the KollRo agent are shown in Figure 7.9.

The process flow for optimising the transport route of the KollRo agent is similar to the process flow of the Manager agent. In the case of the KollRo agent, the ‘Statechart 1’ is active as soon as a workstation in the Main agent, for example workstation WS 1.1, sends a transport order to the KollRo agent. A transport order is sent to the KollRo agent, if the minimum stock of a C part at the workstation is reached.

The KollRo agent waits in the ‘waitForOrder’ state until it receives a transport order from a workstation via the Main agent or by the KollRo agent itself. The creation of a new transport order can have different reasons such as a new order for a C part or due to a failure of the KollRo.

As soon as a new transport order arrives, the Manager agent is triggered by a ‘message’ and transits to the next state. In the ‘initializePopulation’ state, the sequence of the transport orders is randomly created. This is done for all existing chromosomes in the population, thus creating a valid initial population based on the actual transport orders. After generating the initial population, the ‘condition’ trigger is set to true and the KollRo agent moves to the ‘selectChromosome’ state.

In the ‘selectChromosome’ state, the individual chromosomes and their solution value are calculated. To determine the solution value, the KollRo agent locates its current position and its earliest possible transport start. From the TransportOrder agent, the KollRo agent receives information about its source and sink, as well as its desired delivery time. After the selection of the chromosome based on its fitness value, the ‘condition’ trigger is set to true and the KollRo agent moves to the ‘mutateChromosome’ state.

The selected chromosome is mutated in the ‘mutateChromosome’ state. The mutation process is analogous to the process described in Section 6.4.5.1. After the mutation operator is executed, the worst existing solution before the event of the mutation is selected, deleted and replaced by the new, mutated solution. The new population is then re-evaluated and mutated until the termination criterion is reached. As soon as the termination criterion is met, the KollRo agent passes the ‘branch’ transition and moves to the next state.

The crossover operator follows the mutation of the individual chromosomes and their search for a better solution. This process takes place in the ‘crossoverChromosome’ state. The crossover operator is carried out according to the process described in the Section 6.4.5.2. As soon as the termination criterion is met, the Manager agent passes the ‘branch’ transition and moves to the state ‘sendOrderKollRo’. In this state ‘Statechart 2’ is activated to execute the

transport order based on the sequence plan. The ‘condition’ trigger is set to true and the KollRo agent returns to its initial state ‘waitForOrder’ and waits for a new transport order.

‘Statechart 2’ is similar to the statecharts of the agents of the other means of transport. In the ‘Statechart 2’, the KollRo agent waits in the ‘moveToHomeOfKollRo’ state until it receives a transport order from ‘Statechart 1’. The KollRo agent is triggered by a message and transits to the ‘branch’ transition. At the ‘branch’ transition the KollRo agent checks for available transport orders. If the KollRo agent has a transport order to fulfil, the required information about source and sink are stored and the KollRo agent moves to the ‘pickUpOrder’ state. If there is no transport order to be transported, the KollRo agent transits back to the ‘moveToHomeOfKollRo’ state and waits for the next transport order. At the ‘pickUpOrder’ state the KollRo locates its own position in the production environment and calculates its route to the location of the transport order. After the localisation process, the KollRo moves to the pick-up location of the transport order. Once the KollRo arrives at the pick-up location, the KollRo agent is triggered by an ‘agent arrival’ transition and the KollRo agent transits to the ‘loading’ state. The KollRo agent resides at the ‘loading’ state for a certain time to simulate the loading process. Then, the ‘time out’ transition gets activated and the KollRo agent moves to the next state.

As long as the KollRo agent resides in the ‘dropOffOrder’ state, the means of transport moves to the desired destination of the transport order. During the transportation process, the KollRo might fail. The event of a failure is programmed by a ‘rate’ transition. As soon as the ‘rate’ transition is active, the KollRo has a failure and stops moving. The KollRo agent transits to the ‘failure’ state. At the ‘failure’ state, the KollRo locates its position and sends the information about its transport order with the updated x- and y-position to ‘Statechart 1’. In the ‘Statechart 1’, the KollRo agent saves this information of the transport order. The KollRo agent calculates a new sequence plan to allocate the transport orders based on the current information. After sending the information to ‘Statechart 1’, a ‘time out’ is triggered and the KollRo agent transits to the ‘delay’ state. The KollRo agent stays in the ‘delay’ state until it is repaired. Then the ‘timeout’ transition is activated and the KollRo agent transits to the ‘branch’ transition. At the ‘branch’ transition the KollRo agent checks again for available transport orders. If the KollRo agent has a transport order to fulfil, the required information about source and sink are stored and the KollRo agent moves to the ‘pickUpOrder’ state. If there is no transport order to be transported, the KollRo agent transits back to the ‘moveToHomeOfKollRo’ state and waits for the next transport order.

If no failure of the KollRo occurs during the transport, the agent is triggered by the ‘agent arrival’ transition, once the KollRo arrives at the delivery station. The KollRo agent moves to the ‘unloading’ state. The KollRo agent resides at the ‘unloading’ state for a certain time to simulate the unloading process. Then, the ‘time out’ transition gets activated and the KollRo agent moves to the ‘branch’ transition and subsequently performs the steps as described above.

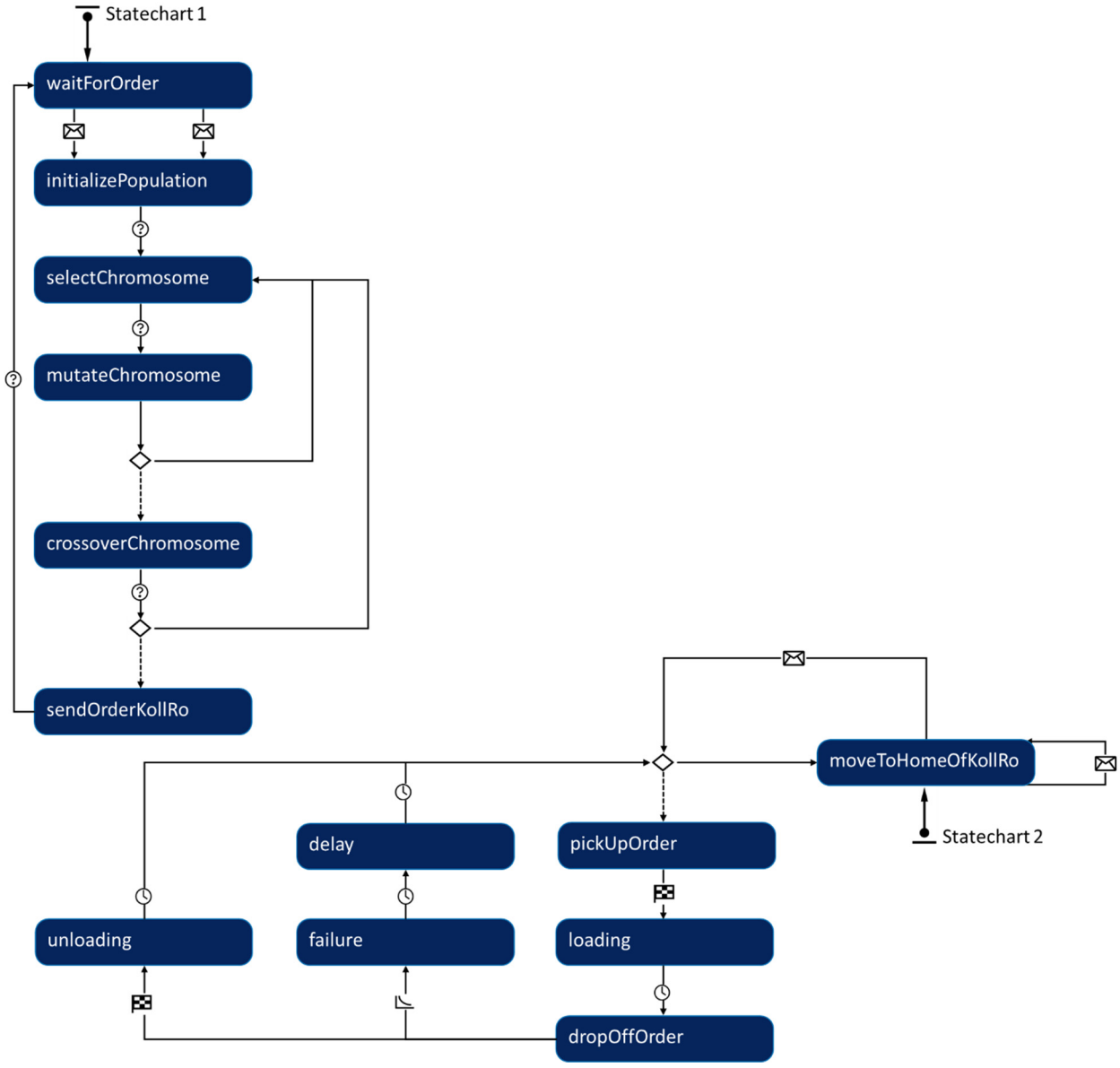


Figure 7.9: Statechart of the KollRo agent

7.3.6 Component agent

The Component agent contains all the necessary information about the A and C parts in this model. The following information about the parts are stored in the Component agent: part type, number of parts per box, storage position in the warehouse and place of consumption. In addition, the consumption of the parts at the various workstations is recorded here. This is important for the supply of the production line. With this information, the required part is reordered at the right time in order to avoid running out of stock. Thus, a new transport order is created and sent to the Manager agent in case of a demand for an A part or to the KollRo agent in case of a demand of a C part.

Chapter 8
Verification and validation of the simulation

V&V techniques should ensure that the results obtained from the simulation study can be used to answer the research questions from Section 1.2. The V&V techniques ensure that the concept and the simulation model meet the theoretical and practical requirements. The basic V&V techniques commonly used in production and logistics are presented. Subsequently, the transfer of verification to the simulation study is described. Finally, the autonomous system is prototypically implemented and validated in the Werk150.

8.1 Description of the verification and validation techniques

There are many different techniques for verifying and validating a simulation model. Therefore, simulation professionals must be able to select the right V&V techniques for a specific phase of a specific simulation study, including the purpose of the study, characteristics of the simulation model, the phase of the simulation study, user knowledge and the availability of data. There is no general approach for selecting the appropriate V&V techniques (Rabe, Spieckermann, & Wenzel, 2008, p. 93). It should be noted that all techniques only serve to reduce errors to a minimum. The usage of V&V techniques leads to an increase in credibility if their application does not indicate an error. In principle, several different techniques should be used to improve the credibility of the simulation model (Rabe et al., 2008, p. 94).

Depending on the phase of the simulation study, different techniques are recommended by the literature. A comprehensive overview of V&V techniques is presented by Rabe et al. (2008, pp. 93–111). Derived from the work of Rabe et al. (2008, pp. 93–111), the V&V techniques applied in this research study were assigned to the phases of the simulation study described in Section 1.4.2.1. A suitable technique should be applied in each phase of the simulation study. According to the simulation study the phases are problem definition, system description, model design, implementation and result analysis. The following section describes the respective V&V techniques used in the present work based on Rabe et al. (2008, pp. 93–111). Figure 8.1 shows the different techniques regarding the different phases of the simulation study.

Animation: Animation can be used to show that the behaviour of a model is valid in certain situations. The temporal processes in the model are represented graphically in 2D or 3D. The animation can only observe whether the processes in the model are logical during the runtime of the simulation and meanwhile detect differences compared to the real system. Errors that rarely occur in the model are unlikely to be detected. The advantage of the animation is to review the model behaviour in selected model sections over short periods.

Trace analysis: The trace analysis tracks the behaviour of individual objects in the executable model, checking logical behaviour and plausibility. The events of the simulation are filtered and evaluated. Typically, a trace file contains continuous information from the model, e.g. model time, identification number and class of an object, location of the object in the model and status of the object and its associated resources.

Extreme condition test: The results of a simulation model must also be logical for combinations of extreme values of the input data and parameters. To perform the extreme-condition test, input variables or parameter values are set in order to make the behaviour of the model more predictable. For example, only a single product is created in the model, thus the number of transport orders in the system is limited to one. As a result, errors during the modelling process can be located more easily, like invalid loading and unloading times or incorrect speed.

Face validity: Validating in dialogue, the model is discussed together with experts who have knowledge of the real system. According to their background, the experts estimate whether the phase results or sections of the model are valid or not. Validation in dialogue can expose errors in case the experts notice discrepancies. Validation in dialogue is also effective because the simulation expert has to explain and rethink his work, thus leading to the detection of errors.

Comparison to other models: The goal of the comparison with other models is to compare the results of the model for certain input data with the results of another, usually simpler model, for the same system with the same input data. However, this technique can only be used if an executable model exists. In addition, statistical techniques can be used to check the correctness of the distribution of input variables in the model. As an example, the determination of the confidence intervals can be used to examine the statistical verification of the initial input data. In order to be able to make reliable statements about the behaviour of the developed model and to achieve trustworthy results, a sufficiently large number of simulation runs with sufficiently long runtimes is generally required.

Event validity test: The event validity test compares the occurrence of events in the simulation

model with the events in reality. For example, the number of orders in a model can be compared to the number of orders in reality. The model is not valid if the type or the number of events does not match the reality with sufficient accuracy.

	Phases of the simulation study				
V & V techniques	Problem definition	System description	Model design	Implementation	Result analysis
Animation				●	
Trace analysis				●	
Extreme-condition test				●	
Face validity	●	●	●	●	●
Comparison to other models				●	
Event validity test				●	

Figure 8.1: V&V techniques used during the simulation study based on Rabe et al. (2008, p. 113)

8.2 Verification of the simulation study

This section describes the implementation of the individual verification techniques during the simulation study. The above-mentioned and theoretically explained techniques are now described, as well as how they have been applied in the simulation study.

8.2.1 Animation

The AnyLogic simulation software offers the modeller the possibility to convert flow diagrams into 2D and 3D graphics. Thereby, a large set of graphical objects can be used to visualise vehicles, employees, buildings and other objects.

During the development of the system, the animation was used to check the model behaviour during runtime. The animation could be used effectively to observe and accurately locate unexpected events during the simulation run. The observation of unexpected events led to a better understanding of the model processes. Thus, errors in the programming could be found systematically. By adjusting the simulation speed, it was possible to follow certain events during the simulation runtime in a more precise way. Thus, a higher transparency and a better observation of the events could be created even in case of multiple events happening simultaneously.

For example, in case of a transport failure, the means of transport is marked by a red circle. Thus, it is very easy for the user to understand why the means of transport suddenly stopped its movement.

Figure 8.2 shows the simulation environment of the Werk150 in 2D and 3D. During the simulation run, it is possible to observe specific areas of interest more closely through intuitive navigation and control, for example by zooming.

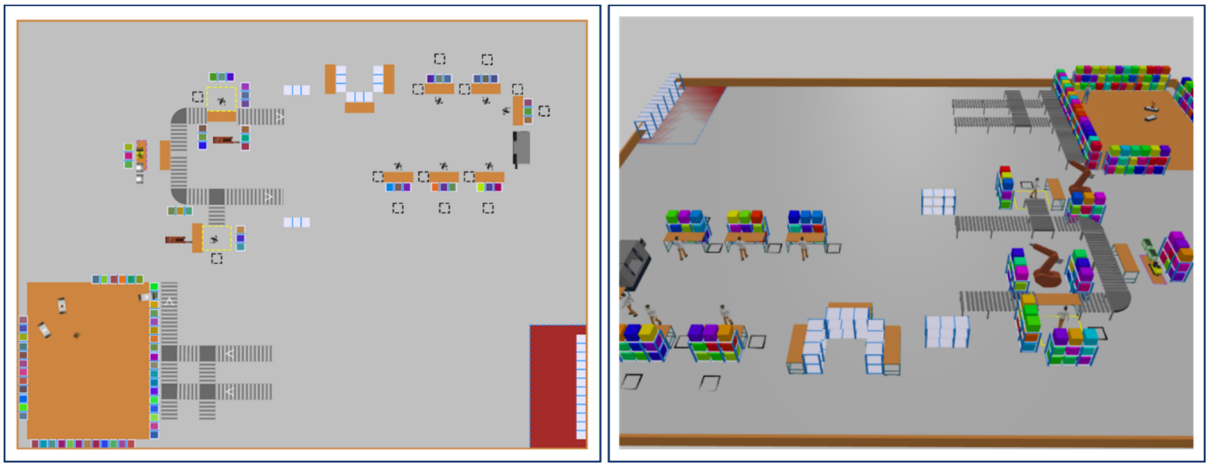


Figure 8.2: 2D and 3D visualisation of the production environment of the Werk150

8.2.2 Trace analysis

AnyLogic provides the user with the traceln() function for textual output information on different occurrences during the model execution. The traceln() function stands for 'trace line' and is primarily used for debugging purposes. With the traceln() function a particular message is printed in the AnyLogic console.

This function was used to check various events during the simulation. For example, the Main agent tracked the creation of a new customer order. With respect to the Manager agent, the various calculated solution values of the population were printed in the console. This ensured that the best possible solution was in fact selected for the order allocation. The means of transport were checked to see whether they were executing the received transport orders or not. In addition, the traceln() function was also used to check the downtimes of the means of transport.

In conjunction with the animation, it is possible to use the trace analysis to detect errors in the simulation model efficiently.

8.2.3 Extreme-condition test

In the extreme-condition test, the system was checked to find if it is also valid in exceptional conditions. The behaviour of the developed system was therefore checked and analysed regarding a simulation scenario where only one city scooter was assembled, limiting the number of orders in the system to 1. Therefore, a feature was programmed, which allows the user to manually create orders. The resulting workload can be calculated and compared with the individual results of the model. In particular, errors can be found in the modelling of process times, e.g. missing conversion factors, invalid loading and unloading times or incorrect speed. Hence, with this technique, the behaviour can be checked very precisely and irregularities in the system can be detected immediately.

8.2.4 Face validity

The validation took place by discussions with the research associates of the Werk150 from the areas of production and logistics as well as information technology. By involving the different departments, the various aspects of the system could be questioned and checked for their coherence. Thereby, it was ensured that the system adequately reflects reality. Weak points identified were eliminated both in the concept and in the implementation phase, so that the overall concept was found to be consistent.

In addition, the validation in dialogue with the research associates was helpful because the simulation had to be explained and rethought. By rethinking the process and procedures in the simulation, inconsistencies and errors could be uncovered.

8.2.5 Comparison to other models

The simulation study aims to prove that the autonomous system has a higher target achievement regarding the logistics target criteria than the static system. Therefore, the two systems are first described regarding their degree of autonomous control. Then, the systems are tested with different parameter configurations. Subsequently, the results are statistically evaluated by comparing the confidence intervals of the systems based on various input variables.

8.2.5.1 Description of the static transport system and the autonomous transport system

In this section, the static transport system and the autonomous transport system are examined regarding to their degree of autonomous control. The criterion catalogue of Böse (2012, p. 25) presented in Section 2.2.2 is used for this purpose. The degree of autonomous control of a transport system is determined by reaching the degree of autonomous fulfilment for each criterion. Figure 8.3 shows the difference of the degree of autonomous control of the static and the autonomous transport system.

Static transport system: The control tasks for allocating the transport orders to the various means of transport are performed centrally for the entire system using a static, global and long-term target system. The process of decision-making is made extrinsically at system level. The logistic objects do not have the ability to communicate or coordinate with each other. The number of decision alternatives for the selection of means of transport is low due to the rules for order allocation. The routes of the individual means of transport are fixed. The sequence in which the transport orders are processed cannot be changed either. The system cannot distinguish between transfer orders with short-term or long-term delivery times and if a means of transport fails, the system cannot react to the new situation. Data are stored and processed primarily by a central IT system. A status measurement of the individual logistic objects with regard to their current position, utilisation time or malfunctions does not take place.

Autonomous transport system: Using the criterion catalogue, the control of the autonomous transport system and the degree of autonomous control are described below in analogy to the static transport system. The organisational structure of the system is mainly heterarchically structured. The control process of allocating transport orders to a means of transport are performed decentrally by the logistics objects. The transport order, the means of transport, the manager and the workstations are involved in the process. The basis for intrinsic decision making at system level is a local target system that is flexible and reacts on short-term changes. The possibility of generating new order sequences and allocating them to the means of transport significantly increases the number of decision alternatives. The transfer from a central instance

to the individual logistics objects leads to redundant decision-making processes and thus to a significant increase of decision possibilities in all decision processes in the system. Since the means of transport are not assigned to a fixed transport order, the allocation of the transport orders to the means of transport is dynamic; hence the system can react to changes. A distinction can be made between the central and the local target systems. For example, the system can distinguish between transfer orders with short-term or long-term delivery times and respond accordingly.

The autonomous allocation of transport orders to the individual means of transport mainly takes place decentrally on the basis of local data and requires a high degree of communication between the logistical objects involved (means of transport, transport orders, workstations). The high communication effort is reflected in a strong increase in data volume compared to the static transport system. The individual logistical objects of the autonomous transport system can also be identified and localised. In addition, the logistical objects have the ability to measure their condition and thus influence the control of the order allocation of the system.

Comparison of the two systems: In order to determine the degree of autonomous control, each criterion is determined by multiplying the corresponding weighting with the fulfilment of the criterion. The value of each criterion is added up to the overall valuation of the system being considered. In total, the static transport system has a degree of autonomous control of 18 points or 17.1 %. Hence, the static system can be considered as an externally controlled transport system.

By multiplying the weighting and the autonomous fulfilment of each criterion and adding up the resulting weighted valuations, the autonomous transport system thus achieves a degree of autonomous control of 78 points or 74.3 %. The developed system can, therefore, be described as autonomous transport system and has a significantly higher degree of autonomous control than the static transport system. The degree of autonomous control determined for the autonomous transport system and the corresponding criteria in the criterion catalogue show that the autonomous transport system still contains elements of more externally controlled systems, such as the distribution of the target system and the place of data processing.

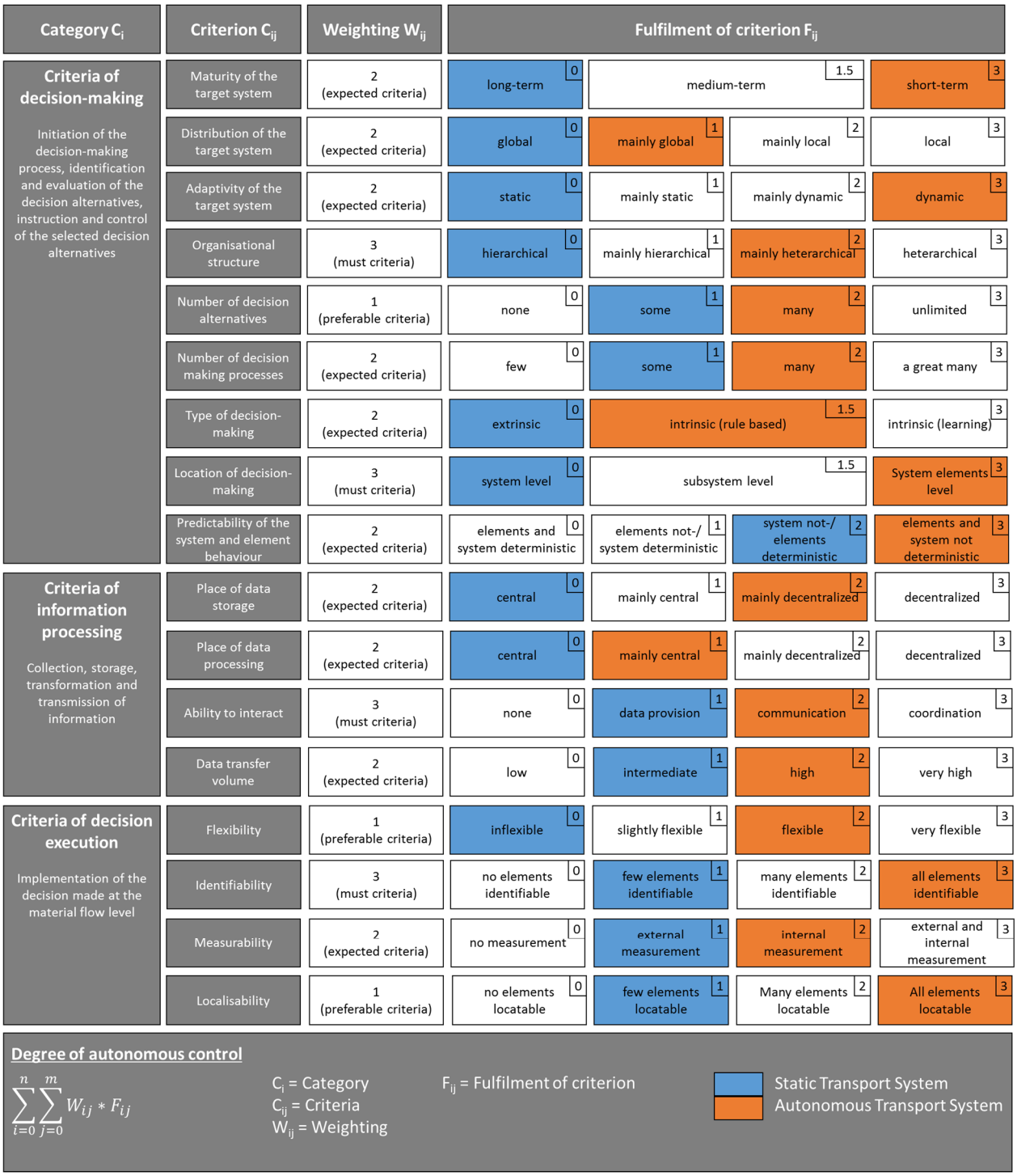


Figure 8.3: Classification of the static and autonomous transport systembased on the criterion catalogue of Böse (2012, p. 25)

8.2.5.2 Statistical evaluation of the simulation results

After the execution of the experiment, the simulation data are available, which can be expected to be statistically valid. Therefore, the data which evaluate the system in terms of their target figures is most relevant. For the evaluation of the system, the target figures described in Section 4.2 are taken into consideration. The evaluation of simulation runs can be differentiated whether they refer to a single parameter configuration or compare different parameter configurations. Concerning this thesis, the comparison of simulation runs with different parameter configurations is discussed in the following paragraphs.

It is not sufficient for a statistically valid evaluation of the two systems to compare the individual mean values of the target criteria. From a statistical point of view, it must be checked, if the two parameter configurations lead to significantly different results. For this work, confidence intervals for the target criteria were calculated in order to analyse the relationship among themselves. If the two intervals do not overlap, the expected values of the two parameter configurations differ statistically in the considered target figure. If the confidence intervals overlap, the evaluation of the data cannot indicate that one expected value is higher than the other (Gutenschwager et al., 2017, pp. 196–197). The calculation of a confidence interval is shown in Appendix E.

For the statistical validation, the data of the simulation runs was stored in an Excel database. As input variable for the simulation runs the parameter ‘time until the next order of a new city scooter’ was chosen. The selected input variables are based on steps of 100 seconds and laying in the range from 100 to 1,000 seconds. Thus, the two systems are tested with ten different input variables. For each input variable, 100 simulation runs were carried out and thus the mean and the confidence intervals of the target figures, defined in Section 4.2, were calculated. The range of input variables was chosen for the following reason. At a range of 100 seconds between the previous customer order and a new customer order, the system is slightly overloaded. At a range of 1,000 seconds between the previous and a new customer order, the system is not working to full capacity. The simulation duration of individual runs was determined statically by a time-related event. The duration of the simulation time was set to 8 hours. The duration ensures that all random events occur often enough so that the underlying random distributions are presented with sufficient accuracy. The data of the simulation runs and the confidence intervals can be found in Appendix E. For each input variable, the confidence intervals for the respective target figure of the static and autonomous system are compared.

A comparison of the confidence intervals of the individual target values leads to the following conclusion. The values of the target figures of the autonomous system are either significantly better than the values of the static system or there is no statistical difference between the two systems. The aim of this simulation study was to verify if an implementation of an autonomous system in terms of the control of intralogistics means of transport fulfils a higher degree of target achievement than a central, statically controlled logistics system. Based on the results of the simulation study, this is the case and the theory from Section 4.1 can be confirmed at a significance level of $\alpha = 0.05$.

8.3 Implementation and validation of the system

Based on the results of the verification procedure discussed in the previous section, it was demonstrated that optimisation potentials exist in intralogistics by using an autonomous control system. Subsequently, the autonomous system is implemented and validated in the Werk150. First, the existing hardware and software required for the implementation of the system in the Werk150 are described. Then, the requirements for a complete implementation of the system are defined, assessed and evaluated. Furthermore, the results of the validation of the autonomous system in the Werk150 are presented.

8.3.1 Description of the software system

For real-time production control, the Self Execution System (SES), which performs typical tasks of a Manufacturing Execution System (MES), is used in the Werk150. The SES, using BECOS oneiroi 2.0 framework, is able to plan production, schedule and execute orders. It offers the opportunity to simulate various production principles and scenarios by taking available orders and resources into consideration. Order prioritisation, communication between all entities based on event-logic and providing operator interfaces with relevant information takes place in the SES (Hummel et al., 2019, p. 352). The system serves as a framework for the development of decentralised control methods within the Werk150. The advantage of the SES, in contrast to conventional MES, is the expandability of the system. Additional resources or services can be integrated and conventional centralised systems or system functionalities can be transferred into the decentralised planning and control of the SES (Schuhmacher & Hummel, 2016, pp. 21–22).

The system architecture of the SES (see Figure 8.4) consists of three interconnected layers, namely ‘administration and configuration’ layer, ‘connectivity’ layer and ‘device’ layer. The ‘administration and configuration’ layer integrates admin functionalities like order management, prioritisation of production orders and specific production control functions as well as services to integrate different kinds of IT systems, infrastructure and resources. The HTTP methods are used to transfer data between the AnyLogic platform and the SES. JavaScript Object Notation (JSON) is used as the data format for transferring structured data in text form. The contents of the transferred data are, for example, information about the source, sink or the desired delivery time of the customer order. The SES controls the ‘connectivity’ layer via the ‘administration and configuration’ layer. In the ‘connectivity’ layer, autonomous software agents can be created for physical objects such as the NeoKu or immaterial objects such as a transport order. These agents provide specific services and can communicate and interact with each other using oneiroi messages and attributes. For the communication between the agents, there are different possibilities from peer-2-peer, where two agents communicate, to multicasting with several agents and finally broadcasting, where all agents are included. Each software agent in the ‘connectivity’ layer has a complementary agent in the ‘device’ layer. This agent is running on or controlling the specific device using APIs such as ROS, OPC-UA and TCP IP interfaces.

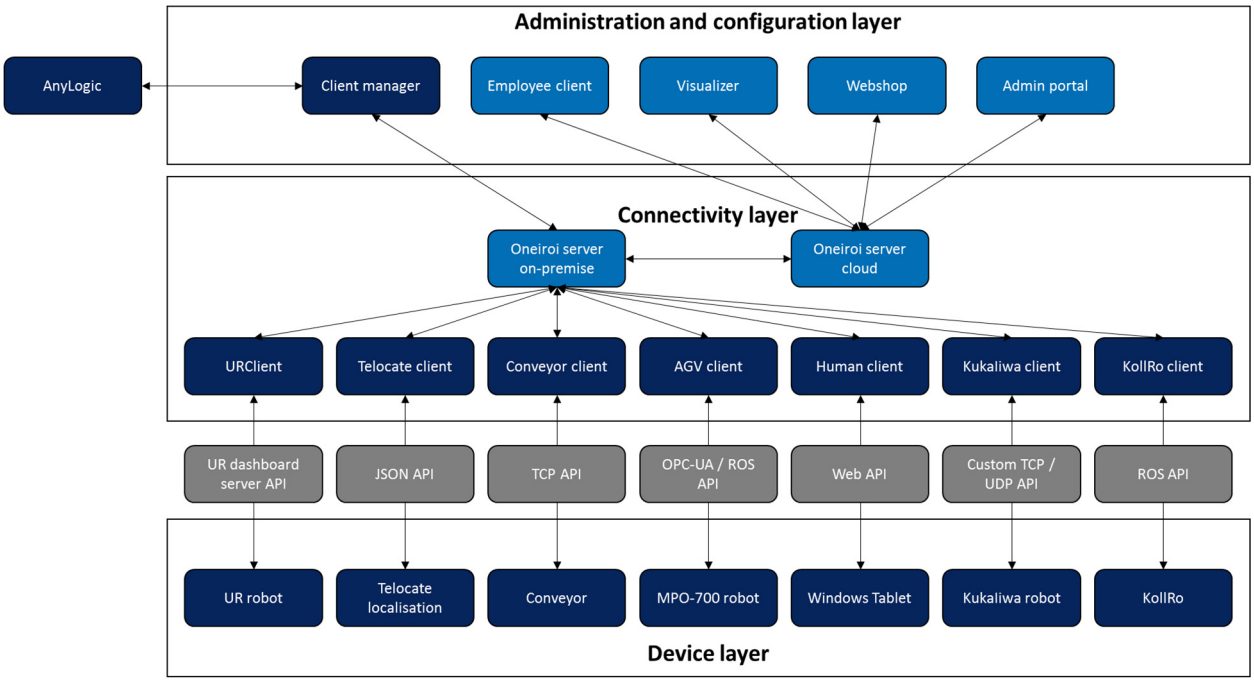


Figure 8.4: System architecture of the SES in the Werk150 (Vijayakumaran Nair, 2019)

8.3.2 Description of the existing hardware

For the integration of the autonomous system into the Werk150, the required hardware is described regarding data processing, indoor-localisation and navigation. Therefore, the intralogistics system at the Werk150 with its different types of means of transport as already described in Section 2.5 as well as the existing indoor navigation system, namely the Telocate Assist localisation system is presented.

AGVs, NeoKu*, KollRo:* The means of transport AGVs, NeoKu and KollRo have the mobile robot MPO-700 from Neobotix as a platform. The robots are equipped with 2D laser scanner Sick S300. This makes it possible to capture the layout of the production and determine the exact position of the transport vehicle. In addition, it is possible for the transport vehicle to independently calculate the transport route and navigate autonomously through the production. For data processing, an Intel Atom based PC with WLAN function is integrated into the transport vehicle. This enables the transport vehicle to receive, send, store and process data. The AGVs, and the NeoKu are running the robot controlled software (PlatformCtrl from Neobotix). The open-source software Robot Operating System (ROS) is installed and configured on the KollRo.

Telocate Assist localisation system: Telocate Assist localisation system is used for the indoor-localisation in the Werk150. The system uses radio or (ultra)sound signals and is connected with the SES. Localisation takes place via a smartphone or tag with integrated loudspeaker by sending an acoustic signal in the range of 18–21 kHz (Hoflinger et al., 2015, p. 2). The installed receivers on the ceiling detect the transmission signal. Depending on the reception times of the transmitted signal at the individual receivers, the position of the smartphone or tag can be calculated. The calculation of the reference positions takes place as a self-calibrating process during operation and the information is sent to the SES (Schuhmacher & Hummel, 2016, pp. 21–22). Figure 8.5 shows the Telocate Assist localisation system at the Werk150 with two active tags.

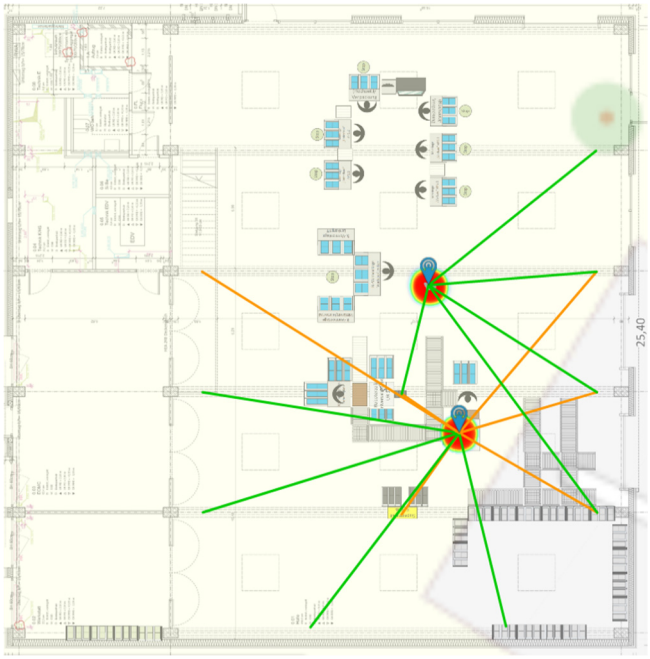


Figure 8.5: Telocate Assist localisation system at the Werk150

Human interface: For the integration of the human as a means of transport, the human is equipped with a tablet and a handcart. The WLAN-capable tablet is using Android as an operating system. With the tablet the human is able to communicate with the other agents and is able to receive, send, store and process data. The tablet can be attached to a handcart, which is provided to the human for the transportation of the city scooter or A parts. Figure 8.6 shows the interface of the human's tablet.

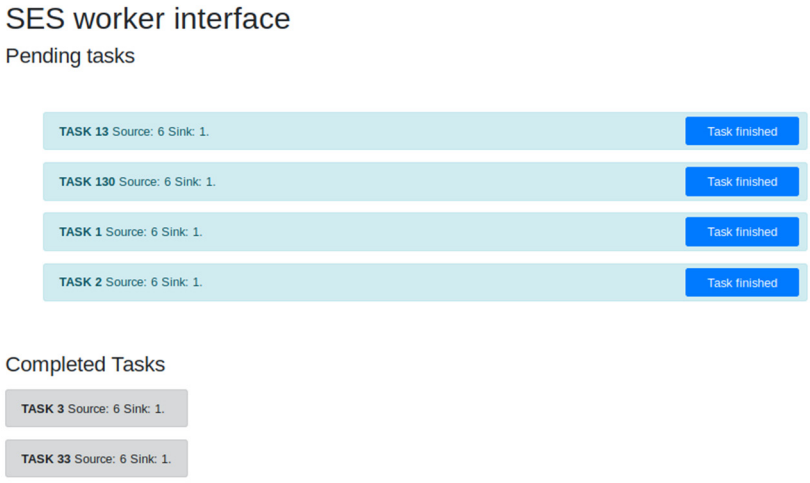


Figure 8.6: Interface of the human's tablet

8.3.3 Implementation of the system in the Werk150

In order to test the successful integration of the developed system into the Werk150, a list of requirements was defined. Table 8.1 describes the six requirements (Req.) for the complete integration of the autonomous system into the Werk150. 'Harvey balls' indicate the degree to which the system was integrated in the Werk150. Following the table, the implementation of the individual requirements is described. A complete implementation of the autonomous system in the Werk150 was not possible due to the relocation to a new building. At the time of implementation and validation, not all hardware systems at the Werk150 were operational.

Table 8.1: Requirements for a complete implementation of the system in the Werk150

Req.	Description of the requirements	Degree of fulfilment
1	Customer orders can be created and imported into the SES.	●
2	City scooters, A parts, C parts and means of transport are locatable in the LFW150.	◐
3	City scooters, A parts and C parts are able to send their current condition to the SES.	◔
4	The logistic target criteria of the system are visualised and measurable.	●
5	The means of transport are able to calculate the transport route and navigate autonomously through production.	●
6	The means of transport can receive and assess transport orders and submit a transport proposal themselves.	◐

○ Unfulfilled ◔ Underperformed ◐ Partially fulfilled ◕ Sufficiently fulfilled ● Fulfilled

Requirement 1: The order of a city scooter, A part or C part with the required information is created in the AnyLogic software. The information about the ID, type of order, source, sink and delivery time are transferred as JSON file to the SES by using HTTP POST request. The SES sends the information of transport orders to the corresponding means of transport. The SES communicates with the robot control software of the Neobotix platforms of the AGVs, NeoKu or the KollRo. The information for the human is sent by using HTTP GET request to a web application on the human's tablet.

Requirement 2: For localisation, each object such as the city scooter, A parts, C parts and the means of transport are equipped with a Telocate tag. The Telocate SES client broadcasts the registered tag position to all SES clients. Due to the relocation of the Werk150 to a new building only two of the Telocate tags were installed and functional for testing at the time of implementation.

Requirement 3: By scanning RFID tags or barcodes of the city scooter, A part or C part, information about ID, type of order, source, sink and delivery time is sent to the SES. The identification of the orders of a city scooter, A parts and the C parts takes place via the integrated RFID tag with the corresponding information. By scanning the city scooter, the processing time for the order can be tracked for each workstation. After the processing time, a new transport order with the information of the order is sent to the SES. Regarding the implementation of the system, it was not possible to introduce RFID tags in the Werk150 yet. Hence, the processing time at the workstations is still based on a methods-time measurement (MTM) for the time being.

Requirement 4: The visualisation of the logistic target figures for the target system of the means of transport and the KollRo is presented in the AnyLogic software. Different diagram types such as line, pie and bar graphs are used for the presentation. The degree of target achievement can thus be tracked continuously during the simulation run. The required information, for example transportation time or utilisation time of the means of transport, is transferred as JSON file from the SES to the AnyLogic software by using HTTP methods. The SES receives its data from the virtual agents in the 'Connectivity' layer.

Figure 8.7 shows an example of the logistical target figures of the target system of the means of transport, AGVs, NeoKu and human (left) and the KollRo (right).

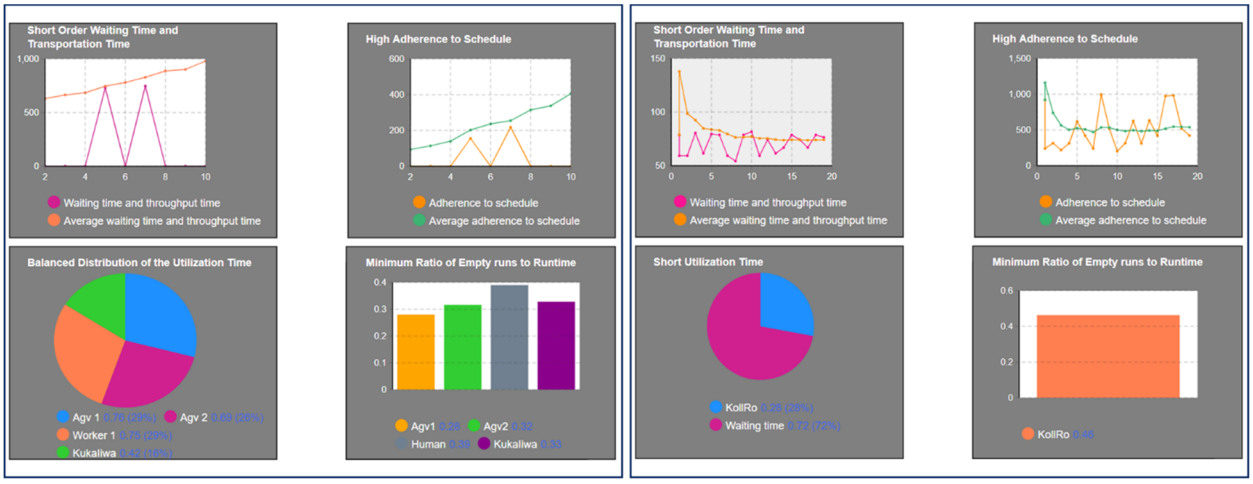


Figure 8.7: Tables of the logistic target criteria of the target system of the means of transport (left) and the KollRo (right)

In addition, the data is also stored in a non-relational database (MongoDB). Hence, the data is also available after the simulation run. The data from the database can be exported to an Excel file and are therefore available for further calculations. The database is shown in Figure 8.8.

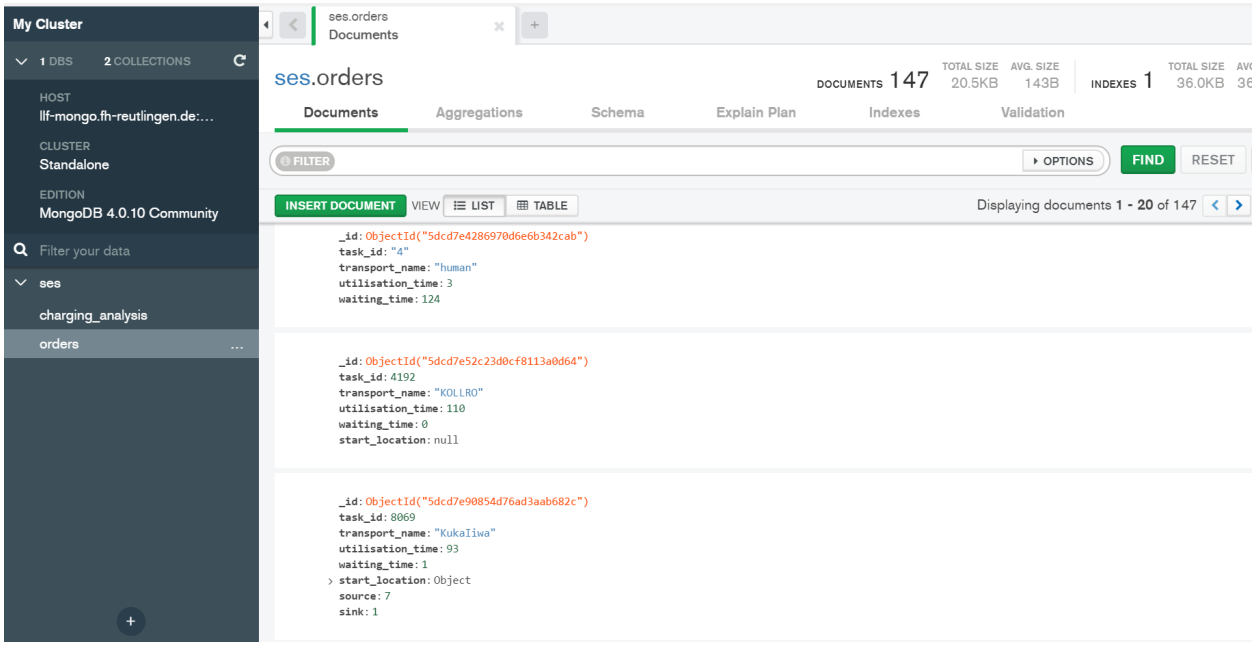


Figure 8.8: Database for storing the data from the SES

Requirement 5: Each transport vehicle receives a virtual representation in the form of an agent in the 'connectivity' layer. Each agent has the properties and states of the associated transport vehicle. The agents receive their transport requests via the SES. These orders contain the information needed to execute the transport order. The transport vehicle is able to locate its own position, to determine the transport route and to move autonomously through the production.

For the localisation, the transport vehicle scans the production hall by using its laser scanner and creates a local map as shown in Figure 8.9 and 8.10. Then the sources and sinks of the transport vehicles are defined in the production layout by using waypoints. The stations of the sources and sinks are reconfigurable through the software. Figure 8.9 shows the layout of the NeoKu with waypoints. The waypoints are the defined sources and sinks, wherein the NeoKu is able to move autonomously.

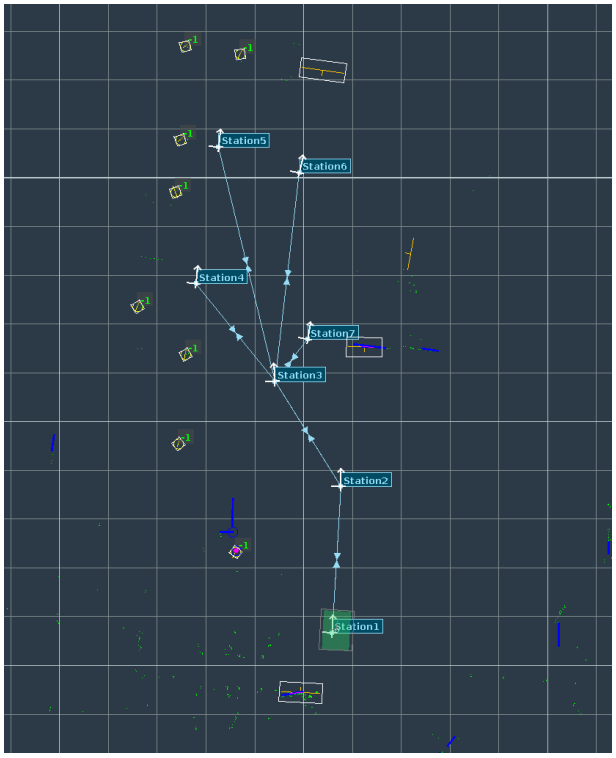


Figure 8.9: Layout of the NeoKu with waypoints visualising accessible sources/sinks

In Figure 8.10 the layout of the KollRo with its respective waypoints is illustrated. The sources and sinks of the KollRo are also defined by waypoints. The figure shows the transport route of the KollRo marked with pink arrows.

Figure 8.10: Layout of the KollRo with waypoints visualising transport route

Requirement 6: With the ROS software the transport vehicles can evaluate the transport orders received and submit their own proposal. The proposal is sent to AnyLogic software via JSON file. The human receives his transport orders on a tablet. Via a web-controlled application, the tablet can process the received transport orders and send them back to the SES. Also, the transport vehicles and the human are able to send their utilisation time and time of empty runs as a JSON file to the SES.

Furthermore, the transport vehicle is able to send its current state to the SES. A transport vehicle can send the states ‘ready’ or ‘in repair’ to the system. If the transport vehicle fails due to technical errors, the status ‘in repair’ is transmitted to the SES as a JSON file. As soon as the transport vehicle is repaired, it sends the status ‘ready’ as a JSON file to the SES. The human using the tablet is not yet able to update its state to the SES.

8.3.4 Practical validation

For the validation of the autonomous system in the Werk150, an event validity test is carried out. The aim is to validate the results obtained from the simulation and to compare them with those of the real system. The validation should ensure that the simulation reflects the behaviour of the real system. For the validation, a total of 100 transport orders were carried out. The gathered data of the Werk150 were recorded in the database and compared with those of the simulation. For statistical analysis, the data was reviewed with a paired Student’s t-test. The results of the test are shown in Appendix E. For the test series, the incoming order value was set at 300, hence an order for a new scooter was loaded into the system every five minutes. The input value ensures that on the one hand the system was still observable during the validation process, on the other hand the number of customer orders loaded into the system was sufficiently high to impose a certain flexibility on the system. In addition, for the event validity test, the validation was carried out without workers. The processing times of the city scooter at the respective station were therefore based on the fixed times of the MTM.

In order to validate the system despite an incomplete implementation of the autonomous system at the Werk150, the events required to calculate and allocate the transport orders to the means of transport must be measurable. In order to apply the event validity test as objectively as possible, hypotheses were formulated before the validation was performed regarding the frequency a particular event would occur and the extent to which the data of the real system have to correspond with the data of the simulation. Therefore, the following events (E) must be considered during the validation test.

E1:	The number of transport orders in the real system is equal to the number of transport orders in the simulation.

First, the number of transport orders in the real system is compared with the number of transport orders in the simulation. The null hypothesis can be accepted if there is no difference in the number of transport orders. The hypothesis that is tested is as follows:

Null Hypothesis 1 *There is no difference between the number of transport orders in the real system and the simulation*

Hypothesis 1 *There is a difference between the number of transport orders in the real system and the simulation*

During the validation phase, 100 transport orders were allocated to the means of transport in the real system and in the simulation. This means that the Null Hypothesis 1 can be accepted in this case.

E2:	Information of source and sink in the real system correspond to the information in the simulation.

Furthermore, the information of the source and sink of the respective transport order must match in the real system and in the simulation. If there are discrepancies in the information, the means of transport would execute the transport order incorrectly. The null hypothesis can be accepted if there is no difference in the information of source and sink of a transport order. The hypothesis that is tested is as follows:

Null Hypothesis 2 *There is no difference between the information of data of source and sink of a transport order in the real system and the simulation*

Hypothesis 2 *There is a difference between the information of data of source and sink of a transport order in the real system and the simulation*

For the observed 100 transport orders, the information about source and sink for the respective transport order was correct, independently of the possible combinations of source and sink. Therefore, the Null Hypothesis 2 can be accepted with regards to the exchange of information.

E3:	The means of transport selected for the respective transport order in the real system corresponds to the means of transport selected in the simulation.

To validate the optimisation process, the allocation of the transport orders to the means of transport in the real system is observed to determine whether it matches the simulation. The null hypothesis can be accepted if there are no differences between the two systems. The hypothesis that is tested is as follows:

Null Hypothesis 3 *There is no difference between the selection of a means of transport for a transport order in the real system and the simulation*

Hypothesis 3 *There is a difference between the selection of a means of transport for a transport order in the real system and the simulation*

The Null Hypothesis 3 could be accepted as there was no difference in the selection of means of transport during the validation phase.

E4:	X- and y-position of the means of transport in the real system is the same as in the simulation at the time of a new transport order entry.

The mean deviation is used to determine the correctness of the x- and y-position of the means of transport at the time of receiving a transport order. Thereby, the x- and y-position of the means of transport in the real system are compared with the x- and y-position of the means of transport in the simulation. The null hypothesis can be accepted if the values of the x- and y-position in the real system do not differ from the simulation by more than 0.5 metres. The difference of 0.5 metres between the two systems is accepted, due to a certain inaccuracy of the Telocate ASSIST localisation system. This difference in the position would also have no significant effect on the allocation of transport orders to the means of transport.. The hypothesis that is tested is as follows:

Null Hypothesis 4 *The difference between the x- and y-position of the means of transport is less than 0.5 metres between the real system and the simulation*

Hypothesis 4 *The difference between the x- and y-position of the means of transport is more than 0.5 metres between the real system and the simulation*

The Null Hypothesis 4 can be accepted since the mean deviation of the x-position is equal to 0.24 metres and for the y-position equal to 0.17 metres. The data of the validation is shown in Appendix E.

E5:	The utilisation time of the means of transport in the real system corresponds to the calculated utilisation time in the simulation.

For the correctness of the utilisation time of the means of transport, the mean deviation of the data is applied as well. The utilisation time of the means of transport in the real system are compared with the calculated utilisation time of the means of transport in the simulation. The null hypothesis can be accepted if the values of utilisation time in the real system do not differ from the simulation by more than 5.0 %. The 5.0 % difference between the two systems is accepted since certain times are not taken into consideration in the simulation for example, the acceleration time of the means of transport. The hypothesis that is tested is as follows:

Null Hypothesis 5 *The difference between the utilisation times of the means of transport is less than 5.0 % between the real system and the simulation*

Hypothesis 5 *The difference between the utilisation times of the means of transport is more than 5.0 % between the real system and the simulation*

Regarding the validation, the Null Hypothesis 5 can be accepted. The values of the utilisation time of the real system and the simulation differ by 4.0 %. The data of the validation is shown in Appendix E.

E6:	The time of empty runs of the means of transport in the real system is the same as the calculated time of empty runs in the simulation.

The mean deviation is used to determine the correctness of the times of empty runs of the means of transports. The times of empty runs of the means of transport in the real system are compared with the calculated times of empty runs of the means of transport in the simulation. The null hypothesis can be accepted if the values of times of empty runs in the real system do not differ from the simulation by more than 5.0. %. This difference is accepted since certain times are not taken into consideration in the simulation, for example, the acceleration time of the means of transport. The hypothesis that is tested is as follows:

Null Hypothesis 6 *The difference between the times of empty runs of the means of transport is less than 5.0 % between the real system and the simulation*

Hypothesis 6 *The difference between the times of empty runs of the means of transport is more than 5.0 % between the real system and the simulation*

The Null Hypothesis 6 can be accepted since the mean deviation of the data of the systems is equal to 4.8 %. The data of the validation is shown in Appendix E.

Chapter 9
Result analysis

In the context of the result analysis, it is necessary to prepare, interpret and evaluate the results of the simulation runs. According to VDI 3633, data is transferred into meaningful key figures that reflect the behaviour of the system and its processes as accurately as possible (Verein Deutscher Ingenieure, 2018, p. 33). Therefore, the logistical behaviour of the regarded system can be described in terms of its target system and logistics target criteria. The target system of the means of transport was evaluated with regards to the four target figures 'high adherence to schedules', 'short order waiting time and transportation time', 'minimum ratio of empty runs to utilisation time' and 'balanced distribution of the utilisation time'. With respect to the target system of the KollRo, the target figures 'high adherence to schedule', 'short order waiting time and transportation time', 'minimum ratio of empty runs to utilisation time' and 'short utilisation time' were evaluated.

Within the result analysis, the system behaviour of the static and the autonomous system is analysed on the basis of the simulation result. The evaluation of the simulation results is based on the original question and objective of the simulation study defined in Section 4.1. Limitations of the model must be taken into account regarding the interpretation of the results (Page, 1991, p. 17). For the interpretation of the individual results, the mean values of the static and autonomous system are compared with each other.

9.1 Target system of the means of transport

In the following section, the average throughput time and the delivery reliability for the order of a new city scooter were determined. In addition, the target figures 'balanced distribution of the utilisation time' and 'minimum ratio of empty runs to utilisation time' for the means of transport were determined. Only the means of transport responsible for transporting a city scooter or A parts to the production line were considered. Hence, the KollRo, which only transports C parts not taken into consideration.

Short order waiting time and transportation time of a city scooter: The comparison of the target figure 'short order waiting time and transportation time of a city scooter' of the two systems is shown in Figure 9.1. Comparing the 'short order waiting time and transportation time' of a city

scooter in the static and the autonomous transport system, it can be noticed that the waiting time and transportation time for the completion of a city scooter is increasing with a high number of orders of a city scooter in the system. Accordingly, the fewer the orders for a city scooter are within the system, the shorter the waiting time and transportation time for the completion of a city scooter will be.

However, the values shown in Figure 9.1 do not decrease towards zero with the decrease of orders of city scooter in the system but remain stable at approximately 820 seconds. The reason for this behaviour of the system can be explained by the fact that approximately 650 seconds are fixed processing and loading and unloading times. It is obvious that the more customer orders entered in the system, the greater the difference between the static system and the autonomous system will be. If a new customer order is created every 100 seconds, the autonomous transport system is on average 692 seconds faster than the static transport system.

In the application scenario where a customer order is loaded into the system every 200 seconds, the mean difference between the two transport systems is 853 seconds, which is the maximum value. The difference in the values of the target figure ‘short order waiting time and transportation time of a city scooter’ between the two systems decreases with increasing time between the creations of a new customer order. At an order interval of 800 seconds, the difference between the values of the two systems is already less than 20 seconds. At an even longer interval per customer order, the static system is slightly better than the autonomous system in terms of the mean order waiting time and transportation time of a city scooter due to the shorter computation time for decision-making. In example, the static system is 2–3 seconds faster than the autonomous system, if only every 900 seconds or 1,000 seconds a new customer order is loaded into the system.

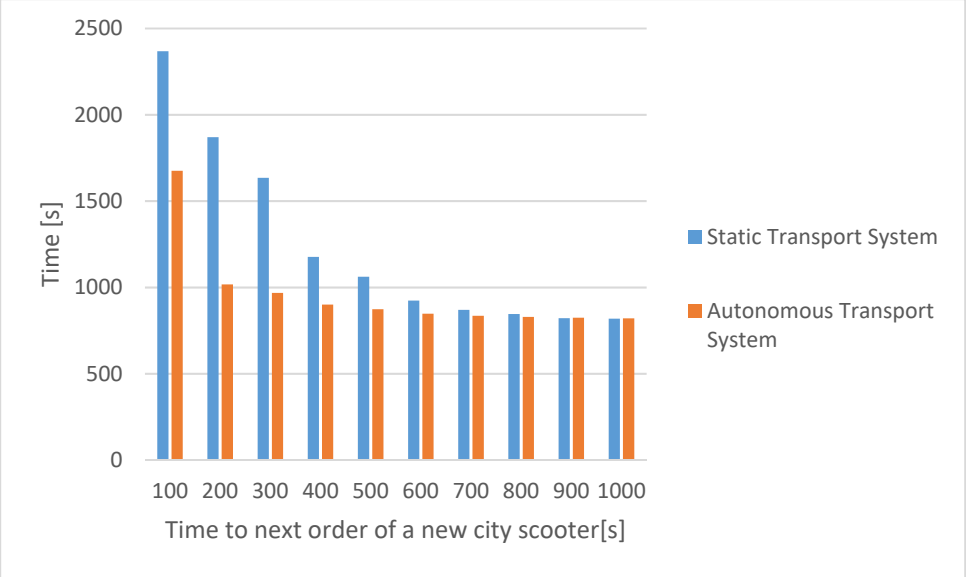


Figure 9.1: Short order waiting time and transportation time of a city scooter

High adherence to schedules of a city scooter: The target figure 'high adherence to schedules' expresses how well the desired delivery time of the customer was met. A high adherence to schedules of the customer orders is of high importance, since completing the city scooter too early will lead to an increase in storage costs; conversely, late completion of the city scooter can lead to penalty costs and/or loss of customers. For the simulation runs, the desired delivery time of a new city scooter varies between 500 and 800 seconds. As already mentioned above, the time for processing, loading and unloading the city scooter until its completion takes 650 seconds. Since the desired delivery time of a new city scooter can also be below 650 seconds, the system can be observed how it reacts to urgent and less urgent customer orders.

Figure 9.2 shows that with a shorter customer order interval, the difference between the time of completion of a city scooter and the desired delivery time for a customer order is growing. The fewer customer orders that were loaded into the system, the better the desired delivery time of the customer can be reached. Analogous to the results shown in Figure 9.1, the autonomous system outperforms the static system especially if the systems have to handle many customer orders. The autonomous system is able to cope with the order quantity more efficiently than the static system in terms of the target figure 'adherence to schedules'. At an interval of 100 seconds until a new customer order is loaded into the system, the autonomous system is on average 685 seconds closer to the desired delivery time than the static system. Creating a new customer order every 200 seconds, the autonomous system outperforms the static system on average by 843 seconds. By increasing the time interval between the old and new customer order, the difference in the values of the two systems decreases. At an interval

of 900 seconds or more, the difference between the static and the autonomous system is less than 10 seconds. If the time to next order of a city scooter is more than 800 seconds, the values of the two systems differ approximately 200 seconds from the desired delivery time. The values of the system then remain stable, the difference between the completion time of a city scooter and the desired delivery time for a customer order does not decrease anymore. This offset is reflecting the discrepancy of the desired delivery time laying between 500 and 800 seconds and processing, loading and unloading time of 650 seconds for the completion of a city scooter, without yet considering waiting and transportation time.

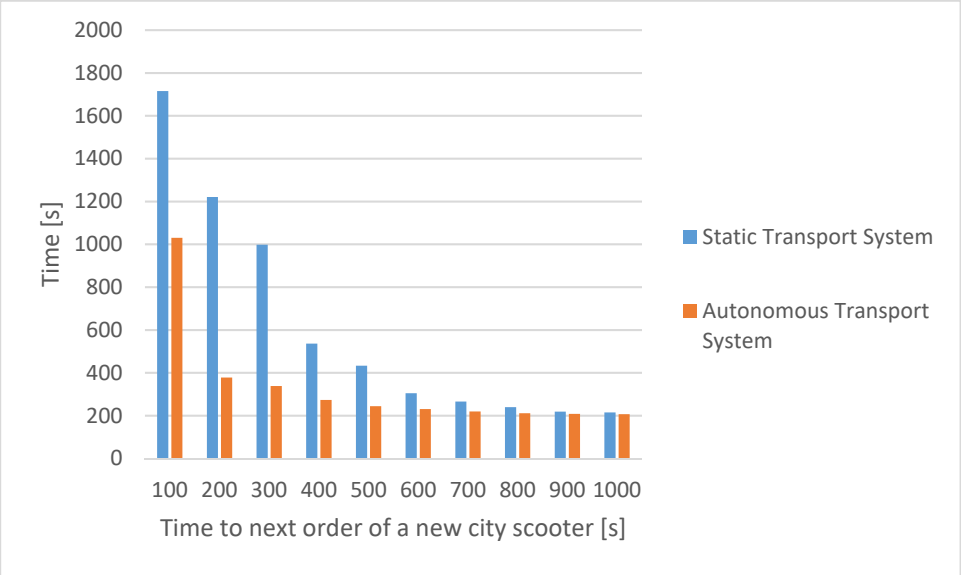


Figure 9.2: Adherence to schedule of the order of a city scooter

Balanced distribution of the utilisation time: The aim of the transport system is to balance the utilisation time of the means of transport. Since the NeoKu can only supply the production line with individual A parts, the NeoKu cannot accept the same transport orders as the AGV or the human. Therefore, the NeoKu has a shorter utilisation time compared to the other means of transport. It is also desirable for the system to reduce the transport tasks of the human. The human would be able to engage in other more sophisticated activities, such as repair of machines or quality checks.

Figure 9.3 shows the distribution of the utilisation time of the individual means of transport in the static system depending on different intervals for the time to the next customer order. The distribution of the utilisation times of the means of transport in a scenario with a high number of customer orders in the system is relatively balanced. Agv 1 received on average 30.0 %, Agv 2 32.8 %, the human 25.9 % and the NeoKu 11.3 % of the transport orders. The fewer customer orders are created in the system, the more transport orders are allocated to the human. On

average, the human received 61.1 % of the transport orders at an interval of 900 and 1,000 seconds between the current and a new customer order in the system. Since the static transport system aims to reduce the order waiting time and transportation time of a city scooter, the means of transport with the fastest transportation time is assigned to transport the order. In scenarios with only a few customer orders, the means of transport are most of the time not assigned to previous transport orders and thus the means of transport with the highest speed is more frequently being selected. Regarding the means of transport of the Werk150, the human with 1.6 m/s is the fastest means of transport, hence the most preferred object.

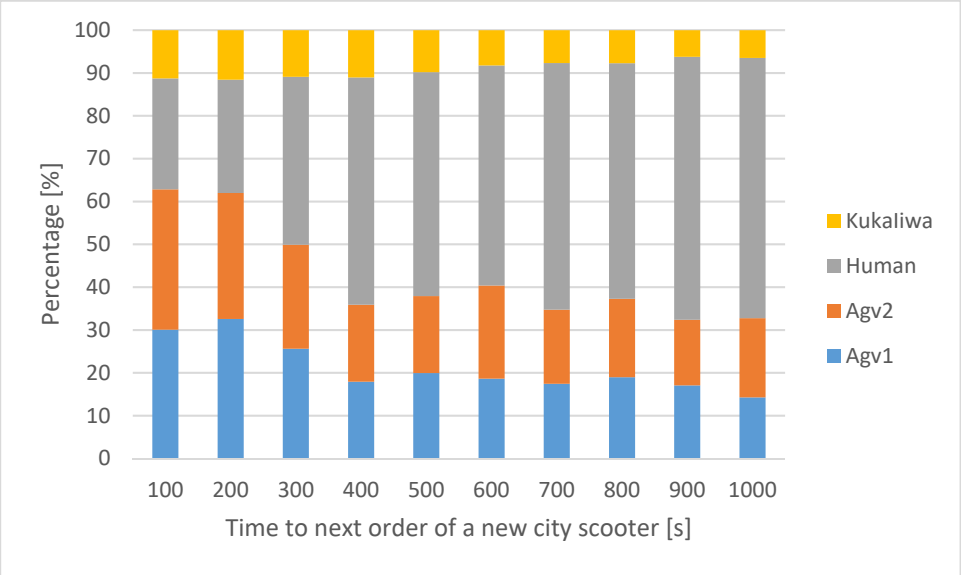


Figure 9.3: Distribution of the utilisation time of the means of transport in the static transport system

The autonomous system also aims to distribute the transport order equally to the means of transport in order to achieve a balanced utilisation time of the means of transport. The distribution of the utilisation time of the individual means of transport in the autonomous system depending on different intervals for the time to the next customer order is shown in Figure 9.4. For example, looking at the distribution of the utilisation time of the means of transport at an interval of 100 seconds per new customer order, AGV 1 receives 27.1 %, AGV 2 25.7 %, the human 32.1 % and the NeoKu 15.1% of the transport orders. Reducing the number of customer orders in the system leads to an increase of assigning transport orders to both AGVs and a reduction of the utilisation time of the human and the NeoKu. At an interval of 1,000 seconds, the AGV 1 receives 40.3 % of the transport orders, the AGV 2 38.7 %, the human 12.5 % and the NeoKu just 8.5 %. Comparing with the static system, the shift in utilisation time is less in the autonomous system.

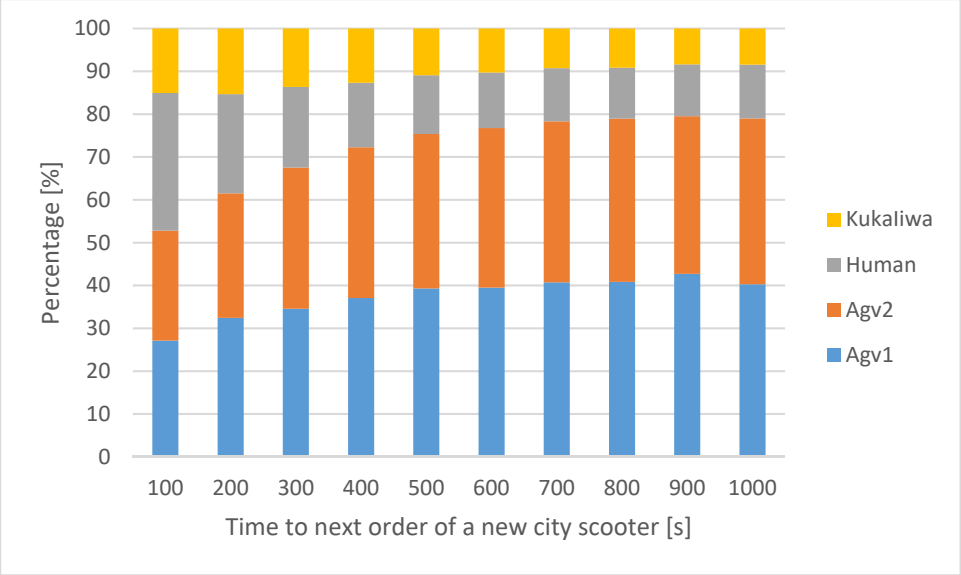


Figure 9.4: Distribution of the utilisation time of the means of transport in the autonomous transport system

Minimum ratio of empty runs to runtime: The transport system aims to reduce the percentage share of empty runs in comparison with the runtime of the means of transport. Figure 9.5 shows that the autonomous system has a smaller proportion of empty runs than the static system, regardless of the current number of customer orders in the system. The percentage share of empty runs at a time interval of 100 seconds is 35.6 % for the static system and 34.9 % for the autonomous system. In case every 1,000 seconds a new customer order is created, the average percentage share of empty runs in the static system is 47.1 % and in the autonomous system 40.8 %. In general, the less customer orders are in the system, the percentage share of empty runs increases. In scenarios with only few customer orders within the system, the means of transport, after finishing a transport order, moves back to the goods receipt and waits for its next transport order. Comparing the percentage shares of the static system with the autonomous system, the less customer orders are in the system, the greater the difference of the percentage shares of empty runs between the two systems. Creating a new customer order every 100 seconds, the percentage share of the empty runs of the static system is only 0.7 percentage points worse than the autonomous system. Creating a new customer order at a time interval of 1,000 seconds, the static system is 6.3 percentage points worse than the autonomous system.

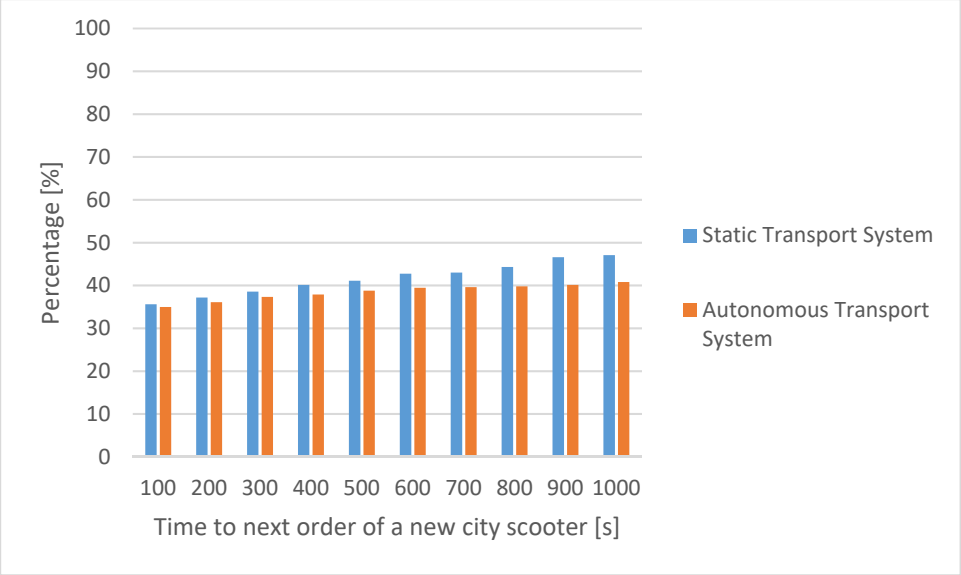


Figure 9.5: Minimum ratio of empty runs to utilisation time

9.2 Target system of the KollRo

The collaborative robot also called KollRo is considered separately with regard to the logistics target criteria since the target figures must be interpreted differently. Analogous to the other means of transport, the target figures were considered with regard to different input variables. In the following, the average throughput time and the delivery reliability for C parts are analysed. In addition, the target figures 'short utilisation time'and 'minimum ratio of empty runs to utilisation time' are interpreted.

Short order waiting time and transportation time of a C part: Starting from the supermarket, the transportation time of a C part to the respective workstation, including loading and unloading times, varies between 60 to 80 seconds. Changing the sequence of the transport orders can lead to an optimisation in the transportation times. However, an unfavourable sequence of the transport orders may lead to a deterioration in the transportation time.

Figure 9.6 shows that the average waiting and transportation time of the KollRo in the static transport system varies between 99 and 109 seconds. The transportation time of the KollRo in the autonomous transport system varies between 72 and 77 seconds. A trend for the values in both systems of the transportation time of the KollRo is not recognisable with respect to the time to the next order. The KollRo in the autonomous system needs on average 30 seconds less transportation time than the KollRo in the static system.

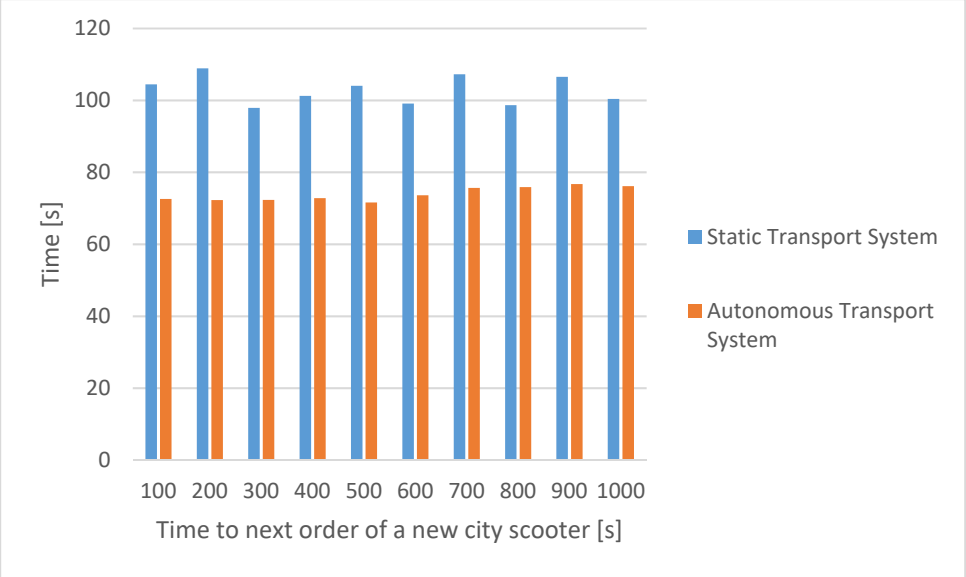


Figure 9.6: Short order waiting time and transportation time of a C part

High adherence to schedules of a C part: The values of difference between the delivery time of a C part at the workstation and the desired delivery time are shown in Figure 9.7 for the static system. The deviations from the desired delivery time to the delivery time of the C part differs by up to 2.5 hours at an interval of 100 seconds for a new customer order. The less transport orders are in the system, the better the KollRo can meet the desired delivery time of a C part in the static system. If every 1,000 seconds a new customer order is loaded into the system, the delivery time of a C part to its desired delivery time differs on average by 48 minutes. However, the large deviation between delivery time and desired delivery time is not an indicator for causing stoppages at the production line. Since the KollRo executes its tour to supply the workstations on a fixed schedule, the C parts usually arrive too early at the workstations.

In contrast to the static system, the KollRo in the autonomous system executes its tour after receiving a transport order. The desired delivery time always depends on the respective C part, since each C part is consumption-controlled. Depending on the desired delivery time of the C part, the KollRo waits with the execution of the transport order to achieve a better adherence to schedules. Figure 9.7 shows that the adherence to schedules of C parts in the autonomous system varies between 14 and 22 minutes. Analogous to the static system, the C parts usually arrive too early, hence not causing stoppages at the production line.

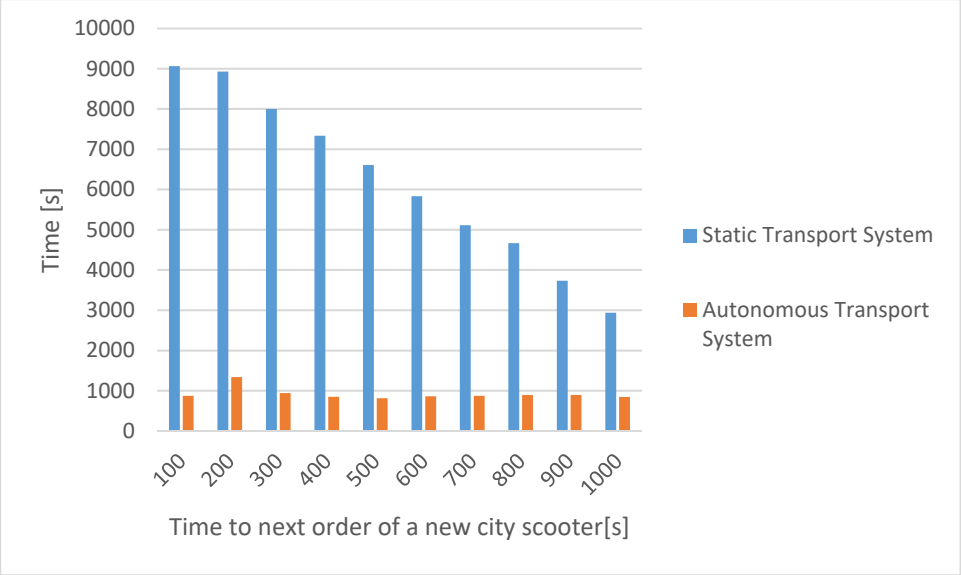


Figure 9.7: High adherence to schedules of a C part

Short utilisation time: The ratio of utilisation time to waiting time of the KollRo should be as low as possible. The utilisation rate of the KollRo itself is not being considered within the simulation study. A lower ratio of utilisation time to waiting time of the KollRo implies that the KollRo could theoretically accept more transport orders. The ratio of utilisation time to waiting time is shown in Figure 9.8 for the static system and in Figure 9.9 for the autonomous system. Both figures show that the KollRo is not fully utilised regardless of the time interval to the next customer order. In this context, it is also understandable that no trend was recognisable for the waiting time and transportation time of an order of a C part in Figure 9.6.

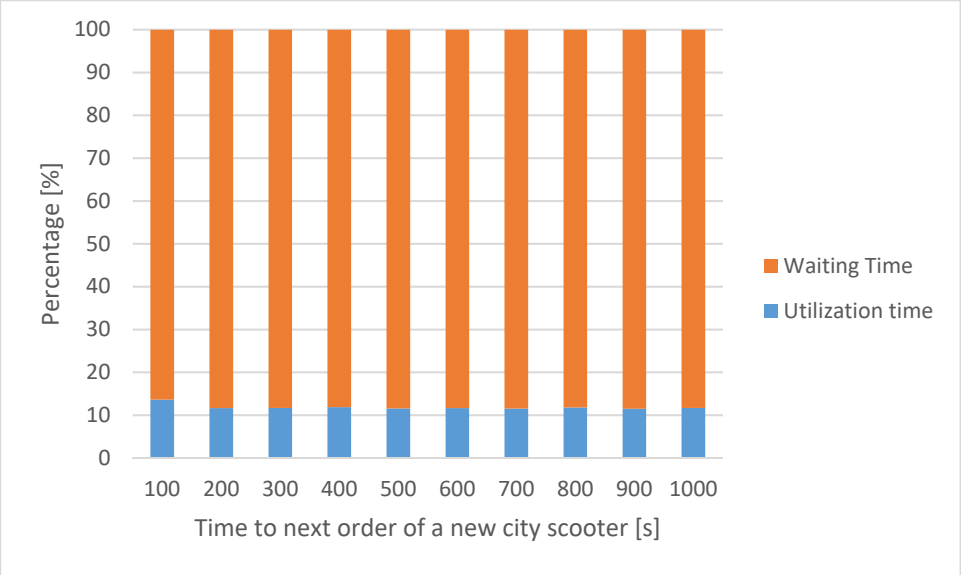


Figure 9.8: Ratio of waiting time and utilisation time of the KollRo in the static transport system

The utilisation time of the KollRo in the static system is relatively balanced and is approximately 11.9 % of the total waiting and utilisation time. Since the KollRo always

executes its tour according to a fixed time schedule, a similar utilisation time for the different intervals is to be expected. Regarding the autonomous system a slight trend of the utilisation time of the KollRo can be recognised. If the number of customer orders in the system is low, the utilisation time of the KollRo is also less. Since the KollRo is consumption-controlled, the delivery of a transport order starts when it receives a transport order from the respective workstation. The utilisation time of the KollRo is on average at 10.2 % at an interval of 100 seconds for a new customer order. The utilisation time of the KollRo decreases to approximately 3.9 % if a new customer order is only loaded into the system every 1,000 seconds.

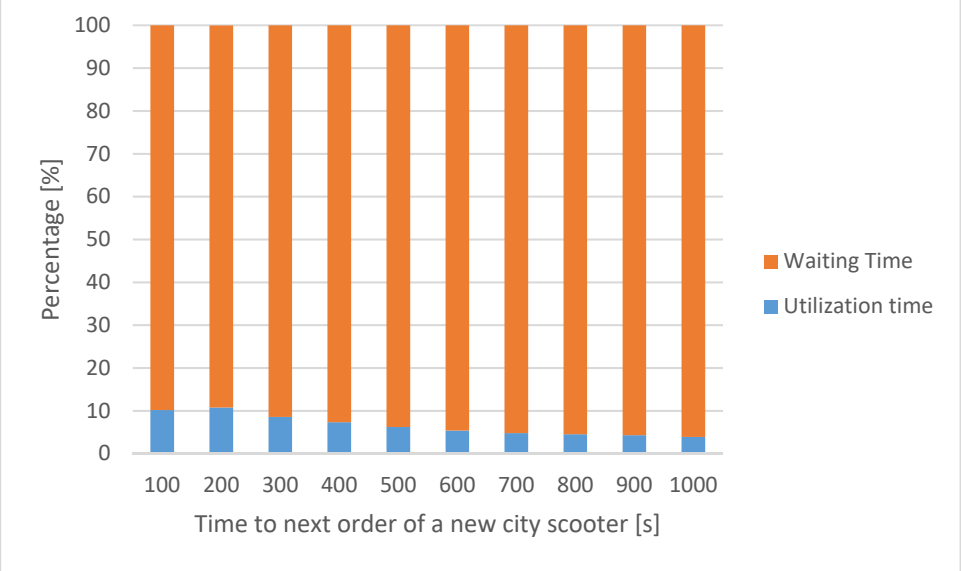


Figure 9.9: Ratio of waiting time and utilisation time of the KollRo in the autonomous transport system

Minimum ratio of empty runs to runtime: Analogous to the other means of transport, the time of the empty runs is set in relation to the runtime. The values of the KollRo in the static and autonomous system are shown in Figure 9.10. The values of both systems do not show any trend which can be explained by the short utilisation time of the KollRo. In the static system, the amount of time of empty runs varies between 89.1 % to 95.9 %. Since the KollRo executes its tour after a fixed time schedule, the KollRo cannot react to the actual demand for C parts. Thus, it is possible that the KollRo does not have to carry out any transport orders during its tour. This explains the high number of empty runs in the static system. Regarding the autonomous system, the KollRo only starts its tour, if a transport order exists. Therefore, the amount of time taken by the empty runs of the KollRo is reduced and varies between 43.0 % and 47.5 %.

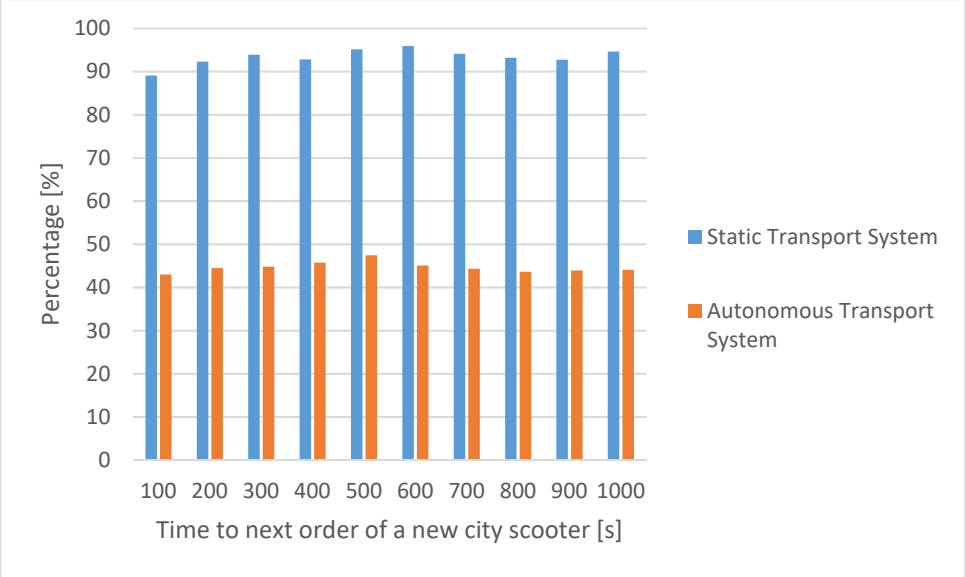


Figure 9.10: Average ratio of empty runs to utilisation time of the KollRo

9.3 Interpretation of the result analysis

This section summarises and interprets the data obtained from the simulation runs of the static and the autonomous system with regard to the two target systems. Furthermore, dependencies between the individual target figures are identified and their relevance in intralogistics transport systems is explained.

Target system of the means of transport: Regarding the target figures of logistics performance 'short order waiting time and transportation time' and 'high adherence to schedules' for the means of transport AGVs, NeoKu and human, the autonomous system achieves much better results than the static system, if a high number of customer orders are in the system. However, the values of the target figures of the logistics performance of the static and autonomous system hardly differ if only few customer orders are loaded into the system. For the target figures of the logistics costs 'balanced distribution of the utilisation time' and 'minimum ratio of empty runs to runtime' the behaviour is different. The values of the static and autonomous system are similar, if the number of customer orders in the system is high. In the scenarios with only few customer orders in the system, the autonomous system achieves better results than the static system.

The values of the static and autonomous system indicate a shift of importance of the logistics target figures depending on the number of customer orders in the system. The target figures of the logistics performance are most important, if a high number of customer orders are loaded into the system. Allocating a high number of customer orders to the means of transport is more difficult than allocating only a few customer orders, since the number of possibilities to allocate

a customer order to a means of transport increases exponentially with an increase in customer orders. Therefore, it is difficult to ensure a high target achievement for the target figures ‘short order waiting time and transportation time’ and ‘high adherence to schedules’. In the scenario with a high number of customer orders in the system, the means of transport are almost or fully utilised with almost no waiting times. Due to the high number of transport orders, the system automatically distributes the transport orders equally to the means of transport. In addition, the means of transport are assigned to at least one transport order in the case of a high number of customer orders in the system. Hence, the amount of time of empty runs of the means of transport is low, since the means of transport are assigned to several transport orders.

The less customer orders there are in the system, the less transport orders have to be fulfilled and the more important the target figures for logistics costs become. With a low number of customer orders, the system aims to distribute the customer orders equally to the means of transport. Since the static system only considers the target figure ‘short order waiting time and transportation time’, the fastest means of transport, the human in the case of the Werk150, will be selected more often than the other means of transport. The static system cannot react to the number of customer orders in the system. The static system can ensure a high target achievement of the target figure ‘short order waiting time and transportation time’ but must accept losses in regards to the other target figures. The amount of time of empty runs increases with decreasing numbers of customer orders in the system. After executing a transport order, the means of transport moves back to the goods receipt. By re-allocating the transport orders, the aim is to reduce the empty runs of the means of transport. The fewer customer orders in the system, the less possibility of combining transport orders exist. Hence, the target figure ‘minimum ratio of empty runs to utilisation time’ and combining transport orders to reduce the empty runs is more important, if the number of customer orders in the system is low. The target figures ‘short order waiting time and transportation time’ and ‘high adherence to schedules’ are less important if only few customer orders are loaded into the system. Since the means of transport are not fully utilised, for each or almost every transport order a means of transport is available. Hence, ensuring a high target achievement of the target figures of logistics performance is to be expected.

In summary, the target figures of the logistics performance of the means of transport AGV, NeoKu and human are of great importance for high numbers of orders in the system; the target figures of the logistics costs are becoming increasingly important in case of small numbers of transport orders in the system.

Target system of the KollRo: The considered target figures and their interpretation for the KollRo differ from the target figure of the other means of transport. The reason for the different approach results from the fact that in the target system of the KollRo only a single means of transport is considered. In general, it should be noted that the KollRo is not fully utilised during the simulation runs independent from the different input variables.

Due to the low utilisation time of the KollRo, a trend cannot be recognised for the target figure 'short order waiting time and transportation time', the values remain stable. Regarding this target figure, the autonomous system achieves a shorter waiting and transportation time by approximately 30 seconds than the static system. Since the KollRo in the autonomous system is able to change the sequence plan based on the current transport orders in the system, the waiting and transportation time of the KollRo can be improved. Hence, the KollRo in the autonomous system is able to ensure a higher degree of target achievement than the KollRo in the static system.

Considering the target figure 'high adherence to schedules', the autonomous system achieves better results than the static system. In the static system, the KollRo executes its transport orders based on a defined time schedule. The desired delivery time of the C part is not taken into account, hence the delivery time can differ considerably from the desired delivery time of the C parts. This is particularly noticeable in scenarios with high numbers of transport orders in the system. The fewer customer orders in the system, the better is the target achievement of the KollRo in the static system. The values in the autonomous system are significantly better than those in the static system. The reason for a higher target achievement of the autonomous system is that the KollRo starts its tour only after it has received a transport order. In addition, the KollRo receives information about the desired delivery time of the C part and calculates the best possible start time for the next transport order.

For the target figure 'short utilisation time' of the static and the autonomous system, the ratio of waiting time to utilisation time is compared. Regarding the static system, the ratio of waiting time to utilisation time remains relatively stable independent of the number of customer orders in the system. Since the KollRo starts its tours according to a defined time schedule, the utilisation time hardly varies. The KollRo in the autonomous system is able to react to the order load in the system. Therefore, the utilisation time in the autonomous system is reduced due to the fewer transport orders the KollRo receives. A short utilisation time of the KollRo ensures a high degree of target achievement, since the KollRo could theoretically accept more transport orders.

In addition, the autonomous system has a higher target achievement than the static system considering the target figure ‘minimum ratio of empty runs to runtime’. Two reasons exist for the autonomous system achieving better results than the static system. On the one hand, the KollRo in the autonomous system is able to change the sequence of the transport orders in order to minimise empty runs, whereas in the static system the KollRo might start its tour without having anything to deliver. In the target system of the KollRo, ensuring a high degree of target achievement is more difficult with a high number of customer orders in the system. The greater the difference in the values of the target figures between the static and the autonomous system, the more transport orders have to be fulfiled by the KollRo. The values of the target figures are similar, if the order load in the system is low.

In contrast to the target system of the other means of transport, there is no shift in the importance of the target figures regarding the target system of the KollRo. A high target achievement of the target figures ‘short order waiting time and transportation time’ and ‘high adherence to schedules’ is more difficult to achieve with a high number of customer orders in the system. In case of unpredictable events, the system must be able to react and, if necessary, change the order sequence to ensure a high target achievement. The possibilities to optimise the system in case of there only being a few customer orders in the system, is limited. This also applies to the target figures ‘short utilisation time’ and ‘minimum ratio of empty runs to runtime’.

In this case, the degree of target achievement of the autonomous system does not differ significantly from that of a static system. Due to the required computation time, the target achievement of the autonomous system could be even worse than the static system. However, due to the low utilisation time of the KollRo in the simulation runs, this could not be identified based on the evaluated data. In this context, (Scholz-Reiter, Beer, Böse, & Windt, 2007) described the limits of autonomous systems, which support the described findings for the target system of the KollRo.

Chapter 10
Summary, conclusion and outlook

Chapter 10 gives the reader an overview of the work discussed in this research study. This chapter comprises three sections. The first section presents a summary of the individual chapters. The findings and conclusions are critically assessed with regard to research objectives and questions in the second section. Finally, the chapter concludes with a discussion of recommendations for further research projects.

10.1 Research summary

Due to saturated markets and a large number of substitution goods in most markets, many companies are forced to be highly flexible when it comes to variants of their products. This demands a high degree of flexibility and responsiveness both from the production system and from the adjoining in-house material supply. As a consequence, logistics must be able to react quickly to disruptions or plan changes. In contrast to conventional logistics systems, autonomous planning and control systems seem to meet today's requirements for flexible and efficient order processing.

The analysis of the literature review in Chapter 3 reveals with regard to the defined objectives of this work (cf. Section 1.3) that existing approaches can only solve these challenges to a limited extent. An examination of existing approaches about control in intralogistics made it possible to derive the need for action with regard to a control system that enables a short-term, autonomous reaction to disturbances considering different target figures. Furthermore, solving realistic VRP with all existing boundary conditions in a very short time needs to be improved.

On the basis of these findings, an autonomous transport system for the control of means of transport in intralogistics was further developed within the scope of the present work in order to optimise the internal material provision within a flexible flow production.

The developed system takes additional restrictions such as the various technical restrictions of the means of transport or external disturbances into account. The developed optimisation algorithm, based on a GA with neighbourhood search, searches the solution space for possible order sequence combinations to generate a high degree of logistical target achievement. By means of a matrix-like representation of the optimisation problem as well as specific operators,

the new solutions are iteratively developed from the initial solution in order to reach a global optimum.

The autonomous system was developed in the software program AnyLogic. For the simulation, the existing production of the Werk150 served as a reference model. For the measurement and evaluation of the logistic goal achievement of the static and the autonomous system, suitable target figures were defined for intralogistics systems. With respect to the target figures defined in Section 4.2, the V&V process shows that the autonomous system ensures a higher logistics target achievement in comparison to the static system. Concerning the autonomous system, waiting and transportation times for a customer order were significantly shortened, by simultaneously improving the adherence to schedules of the customer orders. In addition, a balanced distribution of the orders to the means of transport could be achieved. With respect to the target system of the KollRo, utilisation time could be successfully reduced. The empty runs of the means of transport could also be reduced compared to the static system.

The result analysis of the simulation study (cf. Chapter 9) shows that the use of autonomous systems enables a significant increase in efficiency in intralogistics processes with regard to the logistics target criteria performance and costs.

10.2 Conclusion of the research results

The degree of fulfilment of this research study has been assessed against the objectives defined in Section 1.3. Thereby, the four objectives ‘high efficiency’, ‘holistic view’, ‘flexibility of the overall system’ and ‘real-time capability of the route calculation’ are fulfilled to a high degree.

The objective ‘*High efficiency’* can be regarded as fulfilled, since the autonomous system achieves a higher degree of target achievement of the logistic target values than the static system. The results could be proven during the V&V process. The solution quality proved to be sufficient in theoretical investigations as well as in practice, although the optimal solution cannot be guaranteed due to the use of metaheuristics.

The ‘*Holistic view’* could be fulfilled by the inclusion of various intralogistics means of transport. The considered means of transport differ in regard to their technical requirements, capabilities as well as their degree of flexibility. For the development of the algorithm, it was important not to use any application-specific assumptions or prerequisites in order to be able to ensure applicability to other companies. Since only the current material call-offs and static data such as the production layout are required for operation, the system can be integrated into

almost all conceivable production facilities. The degree of automation of the production is not relevant for the system. The system can be used in highly automated production facilities as well as in production facilities with predominantly manual activities.

The *'Flexibility of the overall system'* could be proven during the tests in simulation and practice. The system was able to react to disturbances during the test trials in order to achieve the highest possible degree of logistical target achievement. The system was able to provide alternative solutions for order processing and thus prove the dynamics and reactivity of the system.

The system can react to new events and, if necessary, change the allocation of the transport orders to the means of transport. The computation time was sufficient for most cases (less than 2 seconds) and proved to be unproblematic during the V&V process. For significantly larger scenarios, it is conceivable to divide the area to be considered in order to reduce the size of the problem and thus reduce the computation time accordingly. Hence, the achievement of the goal *'Real-time capability of the route calculation'* could be proven in the context of the simulation and the practical validation.

In summary, the system developed fulfils most of the requirements defined in Section 1.3, while the other requirement, real-time capability of the route calculation, is met to a sufficient degree. Table 10.1 summarises the fulfilment of the individual requirements.

Table 10.1: Evaluation of the fulfilment of the individual objectives of the thesis

Objectives of the thesis	Degree of fulfilment
High efficiency	●
Holistic view	●
Flexibility of the overall system	●
Real-time capability of the route calculation	◕

○ Unfulfilled ◔ Underperformed ◐ Partially fulfilled ◕ Sufficiently fulfilled ● Fulfilled

10.3 Outlook

A number of limitations have been identified within the present work. However, these limitations provide the opportunity for further research activities to increase the efficiency of intralogistics control. The topics listed below can form the basis for further research projects:

Optimisation of the computation time: The computation time can be further improved by various optimisation measures. For example, the number of possible solutions can be reduced by subdividing the solution space into specific areas without significantly impairing the solution quality. In addition, if there are large optimisation problems, new events can be integrated into the current sequence plan without interrupting the previous sequence plan. Finally, it is conceivable to initially ignore the constraints and only iteratively take them into account in the optimisation process. This procedure, also called relaxation in various algorithms, is also a promising field of research for genetic algorithms with many constraints. All of these improvements do not only improve the computation time but also the scalability of the sequencing problem.

Improvement of forecasting capability: A further field for improvement is the forecasting of future capacity requirements. It should, therefore, be considered if additional forecast data can be recorded by observing patterns such as material consumption in adjacent areas. The method of machine learning can be used to improve intralogistics processes at various points. This is certainly conceivable for the prognosis of incoming orders or for the prognosis of failures depending on system parameters.

It can also be interesting for planning purposes to estimate the entire system behaviour. Based on the forecasts, the demand for capacities could be estimated and controlled accordingly. If the utilisation time of the means of transport is low, the means of transport could be available for other departments.

Validation with fully automated logistics: For a complete validation of the system in the Werk150, the condition monitoring of the city scooter, A parts and C parts will be integrated in the next step. By using RFID tags, the status of the city scooter will be tracked. For the automated monitoring of the stocks of the C parts in the production, the smart bin system described in Schumacher, Baumung, & Hummel (2017) will be integrated into the system.

Furthermore, the total fleet of means of transport, for example the AGVs, will be integrated into the system. The tablet's user interface could also be expanded and improved for a more intuitive operation and display of all necessary information. For example, in addition to

showing general information about the transport order, such as source and sink, the tablet could also display the most efficient route to pick-up and delivery destination.

The possible future research projects proposed in this section represent, from today's point of view, a reasonable way to improve intralogistics control. Due to the major changes in the business environment described in Chapter 1, requirements and their possible solution approaches can change rapidly and may therefore require further research studies in future.

References

Aggteleky, B. (1990). *Fabrikplanung: Werksentwicklung und Betriebsrationalisierung* (Neuausg). München: Hanser.

Ai, T. J., & Kachitvichyanukul, V. (2009). Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. *Computers & Industrial Engineering*, *56*(1), 380–387. https://doi.org/10.1016/j.cie.2008.06.012

AnyLogic (2019). AnyLogic Simulation Software. Retrieved from https://www.anylogic.com/

Arndt, H. (2015). *Logistikmanagement*. Wiesbaden: Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-07212-4

Arora, K., & Arora, M. (2016). Better Result for Solving TSP: GA versus ACO. *International Journal of Advanced Research in Computer Science and Software Engineering*, 219–224.

Aufenanger, M. (2009). *Situativ trainierte Regeln zur Ablaufsteuerung in Fertigungssystemen und ihre Integration in Simulationssysteme* (Dissertation). University of Paderborn, Paderborn.

Axelrod, R. (1997). Advancing the art of simulation in the social sciences. *Complexity*, *3*(2), 16–22. https://doi.org/10.1002/(SICI)1099-0526(199711/12)3:2<16::AID-CPLX4>3.0.CO;2-K

Balseiro, S. R., Loiseau, I., & Ramonet, J. (2011). An Ant Colony algorithm hybridized with insertion heuristics for the Time Dependent Vehicle Routing Problem with Time Windows. *Computers & Operations Research*, *38*(6), 954–966. https://doi.org/10.1016/j.cor.2010.10.011

Bent, R., & van Hentenryck, P. (2006). A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research*, *33*(4), 875–893. https://doi.org/10.1016/j.cor.2004.08.001

Błazewicz, J., Ecker, K., Pesch, E., Schmidt, G., & Weglarz, J. (2007). *Handbook on parallel and distributed processing: From Theory to Applications*. *International handbooks on information systems*. Berlin: Springer. Retrieved from http://www.loc.gov/catdir/enhancements/fy0816/99056742-d.html

Blesing, C., Luensch, D., Stenzel, J., & Korth, B. (2017). Concept of a Multi-agent Based Decentralized Production System for the Automotive Industry. In Y. Demazeau, P. Davidsson, J. Bajo, & Z. Vale (Eds.), *Advances in Practical Applications of Cyber-Physical Multi-Agent Systems: The PAAMS Collection* (pp. 19–30). Cham: Springer International Publishing.

Blum, C., & Sampels, M. (2004). An Ant Colony Optimization Algorithm for Shop Scheduling Problems. *Journal of Mathematical Modelling and Algorithms*, *3*(3), 285–308. https://doi.org/10.1023/B:JMMA.0000038614.39977.6f

Bobrik, A., & Trier, M. (2013). Modellüberblick. In H. Krallmann, A. Bobrik, & O. Levina (Eds.), *Systemanalyse im Unternehmen: Prozessorientierte Methoden der Wirtschaftsinformatik* (6th ed., pp. 73–115). München: Oldenbourg-Verl.

Bochmann, L. S. (2018). *Entwicklung und Bewertung eines flexiblen und dezentral gesteuerten Fertigungssystems für variantenreiche Produkte* (Dissertation). https://doi.org/10.3929/ETHZ-B-000238547

Borshchev, A. (2013). *The big book of simulation modeling: Multimethod modeling with AnyLogic 6*. Chicago: AnyLogic North America.

Böse, F. (2012). *Selbststeuerung in der Fahrzeuglogistik: Modellierung und Analyse selbststeuernder logistischer Prozesse in der Auftragsabwicklung von Automobilterminals*. Zugl.: Bremen, Univ., Diss., 2012. *Informationstechnische Systeme und Organisation von Produktion und Logistik: Vol. 14*. Berlin: GITO-Verl.

Bossel, H. (2004). *Systeme, Dynamik, Simulation: Modellbildung, Analyse und Simulation komplexer Systeme*. Norderstedt: Books on Demand.

Branke, J., & Mattfeld, D. C. (2005). Anticipation and flexibility in dynamic scheduling. *International Journal of Production Research*, *43*(15), 3103–3129. https://doi.org/10.1080/00207540500077140

Brucker, P. (2007). *Scheduling Algorithms*. Dordrecht: Springer. Retrieved from http://gbv.eblib.com/patron/FullRecord.aspx?p=337422

Burduk, A., & Musial, K. (2016). Optimization of Chosen Transport Task by Using Generic Algorithms. In K. Saeed & W. Homenda (Eds.), *Computer Information Systems and Industrial Management* (pp. 197–205). Cham: Springer International Publishing.

Chen, M.-C., Hsiao, Y.-H., Himadeep Reddy, R., & Tiwari, M. K. (2016). The Self-Learning Particle Swarm Optimization approach for routing pickup and delivery of multiple products with material handling in multiple cross-docks. *Transportation Research Part E: Logistics and Transportation Review*, *91*, 208–226. https://doi.org/10.1016/j.tre.2016.04.003

Chien, T. W., Balakrishnan, A., & Wong, R. T. (1989). An Integrated Inventory Allocation and Vehicle Routing Problem. *Transportation Science*, *23*(2), 67–76. https://doi.org/10.1287/trsc.23.2.67

Chmait, N., & Challita, K. (2013). Using simulated annealing and ant-colony optimization algorithms to solve the scheduling problem. *Computer Science and Information Technology*, 208–224.

Conway, R. W., Johnson, B. M., & Maxwell, M. L. (1959). Some Problems of Digital Systems Simulations. *Management Science*, *6*, 92–110.

Cormen, T. H. (2001). *Introduction to algorithms* (2nd ed.). Cambridge Mass.: MIT Press.

Daimler AG (2014). *Geschäfsbericht 2014*. Stuttgart.

Dali, N., & Bouamama, S. (2015). GPU-PSO: Parallel Particle Swarm Optimization Approaches on Graphical Processing Unit for Constraint Reasoning: Case of Max-CSPs. *Procedia Computer Science*, *60*, 1070–1080. https://doi.org/10.1016/j.procs.2015.08.152

Danermark, B. (2002). *Explaining society: Critical realism in the social sciences*. *Critical realism--interventions*. London, New York: Routledge.

Dangelmaier, W. (1999). *Fertigungsplanung: Planung von Aufbau und Ablauf der Fertigung Grundlagen, Algorithmen und Beispiele*. *VDI-Buch*. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-98045-9

Danninger, M. (2012). *Ablaufplanung bei Werkstattfertigung: Ameisenalgorithmen zur Minimierung der mittleren Durchlaufzeit*. Zugl.: Passau, Univ., Diss., 2012 (1. Aufl.). Göttingen: Sierke.

Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science*, *6*, 80–91.

Demo3D (2019). Demo3D - Aweseome Models, Fast. Retrieved from https://www.demo3d.com/

Desrochers, M., Desrosiers, J., & Solomon, M. (1992). A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research*, *40*(2), 342–354. https://doi.org/10.1287/opre.40.2.342

Deutsches Institut für Normung e.V. (2010). *Sicherheit von Maschinen – Anordnung von Schutzeinrichtungen im Hinblick auf Annäherungsgeschwindigkeiten von Körperteilen*. (DIN EN ISO 13855). Berlin: Beuth.

Donati, A. V. [Alberto V.], Montemanni, R. [Roberto], Casagrande, N., Rizzoli, A. E. [Andrea E.], & Gambardella, L. M. [Luca M.] (2008). Time dependent vehicle routing problem with a multi ant colony system. *European Journal of Operational Research*, *185*(3), 1174–1191. https://doi.org/10.1016/j.ejor.2006.06.047

Dorigo, M., & Gambardella, L. M. [Luca Maria] (1997). Ant Colony System: A Cooperative Learning Approach To The Traveling Salesman Problem. *IEEE*, *1*(1).

Easterby-Smith, M., Thorpe, R., & Lowe, A. (2002). *Management research: An introduction* (2. ed.). *SAGE series in management research*. London: SAGE.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In (pp. 39–43). IEEE. https://doi.org/10.1109/MHS.1995.494215

ElMaraghy, H., Schuh, G., ElMaraghy, W., Piller, F., Schönsleben, P., Tseng, M., & Bernard, A. (2013). Product variety management. *CIRP Annals*, *62*(2), 629–652. https://doi.org/10.1016/j.cirp.2013.05.007

Emde, S., & Gendreau, M. (2017). Scheduling in-house transport vehicles to feed parts to automotive assembly lines. *European Journal of Operational Research*, *260*(1), 255–267. https://doi.org/10.1016/j.ejor.2016.12.012

Enterprise Dynamics (2019). Enterprise Dynamics: The leading simulation software platform to assist and support in the modeling and analyzing of virtually any problem. Retrieved from https://www.incontrolsim.com/

Fahrmeir, L., Heumann, C., Künstler, R., Pigeot, I., & Tutz, G. (2016). *Statistik: Der Weg zur Datenanalyse* (8., überarbeitete und ergänzte Auflage). *Springer-Lehrbuch*. Berlin, Heidelberg: Springer Spektrum. https://doi.org/10.1007/978-3-662-50372-0

Feo, T. A., & Resende, M. G.C. (1988). A probalistic heuristic for a computationally difficult set covering problem. *Operations Research*, *8*.

Ferrer, J. M., Ortuño, M. T., & Tirado, G. (2016). A GRASP metaheuristic for humanitarian aid distribution. *Journal of Heuristics*, *22*(1), 55–87. https://doi.org/10.1007/s10732-015-9302-5

Fischer, W., & Dittrich, L. (2004). *Materialfluß und Logistik: Potentiale vom Konzept bis zur Detailauslegung* (2., erweiterte Auflage). *VDI-Buch*. Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-18707-0

Fischetti, M., Toth, P., & Vigo, D. (1994). A Branch-and-Bound Algorithm for the Capacitated Vehicle Routing Problem on Directed Graphs. *Operations Research*, *42*(5), 846–859. https://doi.org/10.1287/opre.42.5.846

Flechtner, H. J. (1984). *Grundbegriffe der Kybernetik: Eine Einführung* (5. Aufl.). *dtv Wissenschaft: Vol. 4422*. München: Deutscher Taschenbuch-Verlag.

Freitag, M., Herzog, O., & Scholz-Reiter, B. (2004). Selbssteuerung logistischer Prozesse - Ein Paradigmenwechsel und seine Grenzen. *Industrie-Management*, 23–27.

Gajpal, Y., & Abad, P. L. (2009). Multi-ant colony system (MACS) for a vehicle routing problem with backhauls. *European Journal of Operational Research*, *196*(1), 102–117. https://doi.org/10.1016/j.ejor.2008.02.025

Gao, L., Zhang, G., Zhang, L., & Li, X. (2011). An efficient memetic algorithm for solving the job shop scheduling problem. *Computers & Industrial Engineering*, *60*(4), 699–705. https://doi.org/10.1016/j.cie.2011.01.003

Gola, A., & Kłosowski, G. (2018). Application of Fuzzy Logic and Genetic Algorithms in Automated Works Transport Organization. In S. Omatu, S. Rodríguez, G. Villarrubia, P.

Faria, P. Sitek, & J. Prieto (Eds.), *Distributed Computing and Artificial Intelligence, 14th International Conference* (pp. 29–36). Cham: Springer International Publishing.

Gonçalves, J. F., Magalhães Mendes, J. J. de, & Resende, M. G.C. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research*, *167*(1), 77–95. https://doi.org/10.1016/j.ejor.2004.03.012

Gotthardt, S., Hulla, M., Eder, M., Karre, H., & Ramsauer, C. (2019). Digitalized milk-run system for a learning factory assembly line. *Procedia Manufacturing*, *31*, 175–179. https://doi.org/10.1016/j.promfg.2019.03.028

Greenwood, A. G. (2016). An Approach to Represent Material Handlers as Agents in Discrete-Event Simulation Models. In J. Bajo, M. J. Escalona, S. Giroux, P. Hoffa-Dąbrowska, V. Julián, P. Novais, . . . R. Azambuja-Silveira (Eds.), *Highlights of Practical Applications of Scalable Multi-Agent Systems. The PAAMS Collection* (pp. 98–109). Cham: Springer International Publishing.

Grijalva, G., Chávez, D., & Camacho, O. (2018). Material Distribution with Mobile Robots in an Industrial Environment: System design and simulation. *International Federation of Automatic Control (IFAC)*, 650–655.

Grzegorz Kłosowski, Arkadiusz Gola, & Thibbotuwawa Amila (2018). Computational Intelligence in Control of AGV Multimodal Systems. *International Federation of Automatic Control (IFAC)*, 1421–1427.

Gunther, R. (2000). Simulation - ein Experiment am digitalen Modell. In K. Feldmann & G. Reinhart (Eds.), *Simulationsbasierte Planungssysteme für Organisation und Produktion: Modellaufbau, Simulationsexperimente, Einsatzbeispiele* (pp. 13–30). Berlin, Heidelberg, s.l.: Springer Berlin Heidelberg.

Günther, H.-O., & Tempelmeier, H. (2003). *Produktion und Logistik* (5., verb. Aufl.). *Springer-Lehrbuch*. Berlin: Springer.

Gutenschwager, K., Rabe, M., Spieckermann, S., & Wenzel, S. (2017). *Simulation in Produktion und Logistik*. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-55745-7

Gyulai, D., Pfeiffer, A., Sobottka, T., & Váncza, J. (2013). Milkrun Vehicle Routing Approach for Shop-floor Logistics. *Procedia CIRP*, *7*, 127–132. https://doi.org/10.1016/j.procir.2013.05.022

Haghani, A., & Jung, S. (2005). A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research*, *32*(11), 2959–2986. https://doi.org/10.1016/j.cor.2004.04.013

Heger, J. (2014). *Dynamische Regelselektion in der Reihenfolgeplanung*. Wiesbaden: Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-658-07982-6

Heppner, F. H., & Grenander, U. (1990). A stochastic nonlinear model for coordinated bird flocks. In S. Krasner (Ed.), *The Ubiquity of chaos* (pp. 233–238). Washington, D.C: American Association for the Advancement of Science.

Hettinger. (1982). *Empfehlung des Bundesministers für Arbeit u. Sozialordnung*, p. 96.

Ho, S. C., & Szeto, W. Y. (2014). Solving a static repositioning problem in bike-sharing systems using iterated tabu search. *Transportation Research Part E: Logistics and Transportation Review*, *69*, 180–198. https://doi.org/10.1016/j.tre.2014.05.017

Ho, W., Ho, G. T.S., Ji, P., & Lau, H. C.W. (2008). A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering Applications of Artificial Intelligence*, *21*(4), 548–557. https://doi.org/10.1016/j.engappai.2007.06.001

Hoflinger, F., Bordoy, J., Simon, N., Wendeberg, J., Reindl, L. M., & Schindelhauer, C. (2015). Indoor-localization system for smart phones. In *IEEE International Workshop 12.10.2015 - 13.10.2015* (pp. 1–6). https://doi.org/10.1109/IWMN.2015.7322974

Hooker, J. N. (1995). Testing heuristics: We have it all wrong. *Journal of Heuristics*, *1*(1), 33–42. https://doi.org/10.1007/BF02430364

Hummel, V. (2014). ESB Logistics Learning Factory: The authentic learning, research and development environment at ESB Business School. Retrieved from https://www.esb-business-school.de/en/research/training-and-research-centre-added-value-and-logistics-systems/research-infrastructure/logistics-learning-factory/

Hummel, V., Ranz, F., & Schuhmacher, J. (2019). Best Practice Example 5: ESB Logistics Learning Factory at ESB Business School at Reutlingen University, Germany. In E. Abele, J. Metternich, & M. Tisch (Eds.), *Learning Factories* (pp. 350–354). Cham: Springer International Publishing.

Hyde, K. F. (2000). Recognising deductive processes in qualitative research. *Qualitative Market Research: An International Journal*, *3*(2), 82–90. https://doi.org/10.1108/13522750010322089

Ignall, E., & Schrage, L. (1965). Application of the Branch and Bound Technique to some Flow-Shop Scheduling Problems. *Operations Research*, 400–412.

Jain, A. S., & Meeran, S. (1998). *A state-of-the-art review of job-shop scheduling techniques*. Dundee.

Kalayci, C. B., & Kaya, C. (2016). An ant colony system empowered variable neighborhood search algorithm for the vehicle routing problem with simultaneous pickup and delivery. *Expert Systems with Applications*, *66*, 163–175. https://doi.org/10.1016/j.eswa.2016.09.017

Kang, P. S., & Bhatti, R. S. (2019). Continuous process improvement implementation framework using multi-objective genetic algorithms and discrete event simulation. *Business Process Management Journal*, *25*(5), 1020–1039. https://doi.org/10.1108/BPMJ-07-2017-0188

Kepaptsoglou, K., Fountas, G., & Karlaftis, M. G. (2015). Weather impact on containership routing in closed seas: A chance-constraint optimization approach. *Transportation Research Part C: Emerging Technologies*, *55*, 139–155. https://doi.org/10.1016/j.trc.2015.01.027

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, *220*.

Kłosowski, G., Gola, A., & Amila, T. (2018). Computational Intelligence in Control of AGV Multimodal Systems. *International Federation of Automatic Control (IFAC)*, 1421–1427.

Korte, B., & Vygen, J. (2018). *Kombinatorische Optimierung*. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-57691-5

Kotzab, H., & Westhaus, M. (2005). *Research methodologies in supply chain management*. Heidelberg, New York: Physica-Verlag. Retrieved from http://site.ebrary.com/lib/alltitles/docDetail.action?docID=10143340

Kousi, N., Michalos, G., Makris, S., & Chryssolouris, G. (2016). Short – term Planning for Part Supply in Assembly Lines Using Mobile Robots. *Procedia CIRP*, *44*, 371–376. https://doi.org/10.1016/j.procir.2016.02.131

Kuka AG (2019). Retrieved from https://www.kuka.com/de-de/produkte-leistungen/robotersysteme/industrieroboter/lbr-iiwa

Kumbharana, S. N., & Pandey, G. M. (2013). A Comparative Study of ACO, GA and SA for Solving Travelling Salesman Problem. *International Journal of Societal Applications of Computer Science*, *2*(2), 224–228.

Labadie, N., Prins, C., & Prodhon, C. (2016). *Metaheuristics for vehicle routing problems*. *Computer engineering series: / coordinated by Nicolas Monmarché and Patrick Siarry ; Volume 3*. London: ISTE Ltd.

Land, A. H., & Doig, A. G. (1960). An Automatic Method of Solving Discrete Programming Problems. *Econometrica*, *28*, 497–520.

Laporte, G. (1992). The Vehicle Routing Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, *59*, 345–358.

Leitão, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, *22*(7), 979–991. https://doi.org/10.1016/j.engappai.2008.09.005

Leung, S. C.H., Zhou, X., Zhang, D., & Zheng, J. (2011). Extended guided tabu search and a new packing algorithm for the two-dimensional loading vehicle routing problem. *Computers & Operations Research*, *38*(1), 205–215. https://doi.org/10.1016/j.cor.2010.04.013

Li, B., Liu, H., Xiao, D., Yu, G., & Zhang, Y. (2017). Centralized and optimal motion planning for large-scale AGV systems: A generic approach. *Advances in Engineering Software*, *106*, 33–46. https://doi.org/10.1016/j.advengsoft.2017.01.002

Lieberoth-Leden, C., Röschinger, M., Lechner, J., & Günthner, W. A. (2017). Logistik 4.0. In G. Reinhart (Ed.), *Handbuch Industrie 4.0: Geschäftsmodelle, Prozesse, Technik* (pp. 451–512). München: Hanser.

Lin, S.-W., Yu, V. F., & Lu, C.-C. (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Systems with Applications*, *38*(12), 15244–15252. https://doi.org/10.1016/j.eswa.2011.05.075

Progenium (2015). *Das Dilemma mit der Vielfalt* [Press release].

Mattern, F., & Mehl, H. [H.] (1989). Diskrete Simulation - Prinzipien und Probleme der Effizienzsteigerung durch Parallelisierung. *Informatik Spektrum*, *12*, 198–210.

Mayring, P. (2015). *Qualitative Inhaltsanalyse: Grundlagen und Techniken* (12., überarb. Aufl.). *Beltz Pädagogik*. Weinheim: Beltz.

McKelvey, B. (1999). Complexity theory in organization science: Seizing the promise or becoming a fad? *Emergence: Complexity and Organization*, *1*.

Mehl, H. [Horst]. (1994). *Methoden verteilter Simulation*. *Programm Angewandte Informatik*. Wiesbaden: Vieweg+Teubner Verlag. https://doi.org/10.1007/978-3-322-90609-0

Mentzer, J., & Kahn, K. (1995). A framework of logistics research. *Journal of Business Logistics*, 231–250.

Micieta, B., Edl, M., Krajcovic, M., Dulina, L., Bubenik, P., Durica, L., & Binasova, V. (2018a). Delegate MASs for coordination and control of one-directional AGV systems: a proof-of-concept. *The International Journal of Advanced Manufacturing Technology*, *94*(1-4), 415–431. https://doi.org/10.1007/s00170-017-0915-8

Micieta, B., Edl, M., Krajcovic, M., Dulina, L., Bubenik, P., Durica, L., & Binasova, V. (2018b). Delegate MASs for coordination and control of one-directional AGV systems: a proof-of-concept. *The International Journal of Advanced Manufacturing Technology*, *94*(1-4), 415–431. https://doi.org/10.1007/s00170-017-0915-8

Montemanni, R. [R.], Gambardella, L. M. [L. M.], Rizzoli, A. E. [A. E.], & Donati, A. V. [A. V.] (2005). Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization*, *10*, 327–343.

Mukhairez, H. H. A., & Maghari, A. Y. A. (2015). Performance Comparison of Simulated Annealing, GA and ACO Applied to TSP. *International Journal of Intelligent Computing Research*, *6*(4), 647–654. https://doi.org/10.20533/ijicr.2042.4655.2015.0080

Musa, R., & Chen, F. F. (2008). Simulated annealing and ant colony optimization algorithms for the dynamic throughput maximization problem. *The International Journal of Advanced Manufacturing Technology*, *37*(7-8), 837–850. https://doi.org/10.1007/s00170-007-1005-0

Neobotix (2019). Mobile Roboter. Retrieved from https://www.neobotix-roboter.de/mobile-roboter-uebersicht.html

Niehues, M. R. (2016a). *Adaptive Produktionssteuerung für Werkstattfertigungssysteme durch fertigungsbegleitende Reihenfogebildung* (Dissertation). TU München, München.

Niehues, M. R. (2016b). *Adaptive Produktionssteuerung für Werkstattfertigungssysteme durch fertigungsbegleitende Reihenfolgebildung* (Dissertation). TU München, München.

Norouzi, N., Sadegh-Amalnick, M., & Alinaghiyan, M. (2015). Evaluating of the particle swarm optimization in a periodic vehicle routing problem. *Measurement*, *62*, 162–169. https://doi.org/10.1016/j.measurement.2014.10.024

Novaes, A. G.N., Bez, E. T., Burin, P. J., & Aragão, D. P. (2015). Dynamic milk-run OEM operations in over-congested traffic conditions. *Computers & Industrial Engineering*, *88*, 326–340. https://doi.org/10.1016/j.cie.2015.07.010

Page, B. (1991). *Diskrete Simulation: Eine Einführung mit Modula-2*. *Springer-Lehrbuch*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Patzak, G. (1982). *Systemtechnik - Planung komplexer innovativer Systeme: Grundlagen, Methoden, Techniken.*

Pawlewski, K., & Anholcer, M. (2019). Using CSP Solvers as Alternative to Simulation Optimization Engines. In P. Pawlewski, P. Hoffa-Dabrowska, P. Golinska-Dawson, & K. Werner-Lewandowska (Eds.), *EcoProduction, Environmental Issues in Logistics and Manufacturing. FlexSim in Academe: Teaching and Research* (pp. 131–143). Cham: Springer International Publishing.

Pawlewski, P., & Anholcer, M. (2019). Using CSP Solvers as Alternative to Simulation Optimization Engines. In P. Pawlewski, P. Hoffa-Dabrowska, P. Golinska-Dawson, & K. Werner-Lewandowska (Eds.), *EcoProduction, Environmental Issues in Logistics and Manufacturing. FlexSim in Academe: Teaching and Research* (pp. 131–143). Cham: Springer International Publishing.

Peter, J.P., & Olson, J. C. (1983). Is Science Marketing? *American Marketing Association*, *47*, 111–125.

Pidd, M. (2004). *Computer simulation in management science* (5. ed.). Chichester: Wiley.

Pino, R., Martínez, C., Villanueva, V., Priore, P., & Fernández, I. (2012). Application of GRASP methodology to Vehicle Routing Problem (VRP). In *Proceedings on the International Conference on Artificial Intelligence (ICAI)*.

Plant Simulation (2019). Simulation mit Plant Simulation. Retrieved from https://www.plant-simulation.de/

Prais, M., & Ribeiro, C. C. (2000). Reactive GRASP: An Application to a Matrix Decomposition Problem in TDMA Traffic Assignment. *Journal on Computing*, *12*.

Rabe, M., Spieckermann, S., & Wenzel, S. (2008). *Verifikation und Validierung für die Simulation in Produktion und Logistik: Vorgehensmodelle und Techniken*. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-35282-2

Rekersbrink, H. (2012). *Methoden zum selbststeuernden Routing autonomer logistischer Objekte: Entwicklung und Evaluierung des Distributed Logistics Routing Protocol (DLRP)* (Dissertation). University of Bremen, Bremen.

Rekersbrink, H., Scholz-Reiter, B., & Zabel, C. (2010). An autonomous control concept for problem logistics. In W. Dangelmaier (Ed.), *Lecture notes in business information processing: Vol. 46. Advanced manufacturing and sustainable logistics: 8th international Heinz Nixdorf Symposium, IHNS 2010, Paderborn, Germany, April 21-22, 2010. proceedings* (1st ed., pp. 245–256). Berlin, New York: Springer.

Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. *Computer Graphics*, *21*, 25–34.

Rieck, J., & Zimmermann, J. (2010). A new mixed integer linear model for a rich vehicle routing problem with docking constraints. *Annals of Operations Research*, *181*(1), 337–358. https://doi.org/10.1007/s10479-010-0748-4

Ropohl, G. (2009). *Allgemeine Technologie : eine Systemtheorie der Technik*. s.l.: KIT Scientific Publishing.

Saunders, M. N. K., Lewis, P., & Thornhill, A. (2009). *Research methods for business students* (5th ed.). New York: Prentice Hall.

Schlott, S. (2005). Schlott_2005_Wahnsinn mit Methode, 38–42.

Scholz, D. (2010). *Innerbetriebliche Standortplanung: Das Konzept der Slicing Trees bei der Optimierung von Layoutstrukturen*. *Gabler Research*. Wiesbaden: Gabler Verlag / GWV Fachverlage GmbH Wiesbaden. Retrieved from http://dx.doi.org/10.1007/978-3-8349-8679-5 https://doi.org/10.1007/978-3-8349-8679-5

Scholz-Reiter, B., Beer, C. d., Böse, F., & Windt, K. (2007). Evolution in der Logistik: Selbststeuerung logistischer Prozesse. In Deutscher Materialfluss-Kongress (Ed.), *16. Deutscher Materialfluss-Kongress : Intralogistik - Innovation und Praxis* (pp. 179–190). Düsseldorf: VDI Verlag.

Scholz-Reiter, B., Hildebrandt, T. [Torsten], & Tan, Y. (2013). Effective and efficient scheduling of dynamic job shops—Combining the shifting bottleneck procedure with variable neighbourhood search. *CIRP Annals*, *62*(1), 423–426. https://doi.org/10.1016/j.cirp.2013.03.047

Scholz-Reiter, B., Windt, K., & Freitag, M. (Eds.) (2004). *Autonomous Logistic Processes - New Demands and First Approaches.*

Scholz-Reiter, B., Windt, K., Kolditz, J., Böse Felix, Hildebrandt, T., Philipp, T., & Höhns, H. (Eds.) (2005). *New Concepts of Modelling and Evaluating Autonomous Logistics Processes*. Bremen.

Schuhmacher, J., Baumung, W., & Hummel, V. (2017). An Intelligent Bin System for Decentrally Controlled Intralogistic Systems in Context of Industrie 4.0. *Procedia Manufacturing*, *9*, 135–142. https://doi.org/10.1016/j.promfg.2017.04.005

Schuhmacher, J., & Hummel, V. (2016). Decentralized Control of Logistic Processes in Cyber-physical Production Systems at the Example of ESB Logistics Learning Factory. *Procedia CIRP*, *54*, 19–24. https://doi.org/10.1016/j.procir.2016.04.095

Schyns, M. (2015). An ant colony system for responsive dynamic vehicle routing. *European Journal of Operational Research*, *245*(3), 704–718. https://doi.org/10.1016/j.ejor.2015.04.009

Shchekutin, N., Sohrt, S., & Overmeyer, L. (2017). Multi-objective layout optimization for material flow system with decentralized and scalable control. *Logistics Journal.* Advance online publication. https://doi.org/10.2195/LJ_PROC_SHCHEKUTIN_EN_201710_01

Silberholz, J., Golden, B., Gupta, S., & Wang, X. (2019). Computational Comparison of Metaheuristics. In M. Gendreau & J.-Y. Potvin (Eds.), *Handbook of Metaheuristics* (pp. 581–604). Cham: Springer International Publishing.

Silva, C. A., Sousa, J.M.C., & Runkler, T. A. (2008). Rescheduling and optimization of logistic processes using GA and ACO. *Engineering Applications of Artificial Intelligence*, *21*(3), 343–352. https://doi.org/10.1016/j.engappai.2007.08.006

Simul8 (2019). Process simulation software for fast, confident decision-making. Retrieved from https://www.simul8.com/

Sohrt, S., Heinke, A., Shchekutin, N., Eilert, B., Overmeyer, L., & Krühn, T. (2017). Kleinskalige, cyber-physische Fördertechnik. In B. Vogel-Heuser, T. Bauernhansl, & M. ten Hompel (Eds.), *Springer Reference Technik. Handbuch Industrie 4.0: Bd. 3: Logistik* (2nd ed., pp. 21–44). Berlin: Springer Vieweg.

Stachowiak, H. (1973). *Allgemeine Modelltheorie*. Wien: Springer.

Stentoft Arlbjørn, J., & Halldorsson, A. (2002). Logistics knowledge creation: reflections on content, context and processes. *International Journal of Physical Distribution & Logistics Management*, *32*(1), 22–40. https://doi.org/10.1108/09600030210415289

Stepanenko, S. (2008). *Global optimization methods based on Tabu search* (Dissertation). Julius-Maximilians-Universität Würzburg, Würzburg.

Stewart, W. J. (2009). *Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling*. New Jersey: Princeton University Press.

Subramanian, A., Penna, P. H. V., Uchoa, E., & Ochi, L. S. (2012). A hybrid algorithm for the Heterogeneous Fleet Vehicle Routing Problem. *European Journal of Operational Research*, *221*(2), 285–295. https://doi.org/10.1016/j.ejor.2012.03.016

Suhl, L., & Mellouli, T. (2013). *Optimierungssysteme*. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-38937-5

Szymon, J., & Dominik, Ż. (2013). Solving Multi-criteria Vehicle Routing Problem by Parallel Tabu Search on GPU. *Procedia Computer Science*, *18*, 2529–2532. https://doi.org/10.1016/j.procs.2013.05.434

Teschemacher, U. (2019). *Dynamische Routenzugoptimierung bei kurzfristigen Materialabrufen* (Dissertation). Technische Universität München, Munich.

Teschemacher, U., & Reinhart, G. (2017). Ant Colony Optimization Algorithms to Enable Dynamic Milkrun Logistics. *Procedia CIRP*, *63*, 762–767. https://doi.org/10.1016/j.procir.2017.03.125

Trier, M., Bobrik, A., Neumann, N., & Wyssussek, B. (2013). Systemtheorie und Modell. In H. Krallmann, A. Bobrik, & O. Levina (Eds.), *Systemanalyse im Unternehmen: Prozessorientierte Methoden der Wirtschaftsinformatik* (6th ed., pp. 41–72). München: Oldenbourg-Verl.

Verein Deutscher Ingenieure (2018). *Simulation von Logistik-, Materialfluss- und Produktionssystemen*. (VDI, VDI 3633). Düsseldorf: Beuth.

Vidal, T., Crainic, T. G., Gendreau, M., & Prins, C. (2013). A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, *40*(1), 475–489. https://doi.org/10.1016/j.cor.2012.07.018

Vijayakumaran Nair, V. (2019). *Self Execution System*. Reutlingen.

Walenta, R., Schellekens, T., Ferrein, A., & Schiffer, S. (2017). A Decentralised System Approach for Controlling AGVs with ROS. *IEEE Africon 2017 Proceedings*, 1436–1441.

Wang, C., Mu, D., Zhao, F., & Sutherland, J. W. (2015). A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows. *Computers & Industrial Engineering*, *83*, 111–122. https://doi.org/10.1016/j.cie.2015.02.005

Wenger, W. (2010). *Multikriterielle Tourenplanung*. Zugl.: Hohenheim, Univ., Diss., 2009 (1. Aufl.). *Information, Organisation, Produktion*. Wiesbaden: Gabler. Retrieved from http://dx.doi.org/10.1007/978-3-8349-8613-9 https://doi.org/10.1007/978-3-8349-8613-9

Wiendahl, H.-P. (2008). Produktionscontrolling. In D. Arnold, H. Isermann, A. Kuhn, H. Tempelmeier, & K. Furmans (Eds.), *Handbuch Logistik* (pp. 361–387). Berlin, Heidelberg: Springer Berlin Heidelberg.

Wiendahl, H.-P. (2010). *Betriebsorganisation für Ingenieure* (7., aktualisierte Aufl.). München: Hanser. Retrieved from http://www.hanser-elibrary.com/isbn/9783446418783 https://doi.org/10.3139/9783446422889

Wiendahl, H.-P., Nyhuis, P., & Grabe, D. (2007). Controlling in Lieferketten. In G. Schuh (Ed.), *VDI-Buch. Produktionsplanung und -steuerung: Grundlagen, Gestaltung Und Konzepte* (pp. 467–510). Dordrecht: Springer.

Windt, K. (2008). Ermittlung des angemessenen Selbststeuerungsgrades in der Logistik – Grenzen der Selbststeuerung. In P. Nyhuis (Ed.), *Beiträge zu einer Theorie der Logistik* (pp. 349–372). Berlin, Heidelberg: Springer Berlin Heidelberg.

Windt, K., & Hülsmann, M. (2007). Changing Paradigms in Logistics – Understanding the Shift from Conventional Control to Autonomous Cooperation and Control. In K. Windt & M. Hülsmann (Eds.), *Understanding autonomous cooperation and control in logistics: The impact of autonomy on management, information, communication and material flow* (pp. 1–12). Berlin, New York: Springer.

Wunsch, G., & Schreiber, H. (2006). *Stochastische Systeme* (4., neu bearb. Aufl.). Berlin, Heidelberg: Springer-Verlag. https://doi.org/10.1007/3-540-29226-8

Yu, B., Yang, Z. Z., & Yao, B. Z. (2011). A hybrid algorithm for vehicle routing problem with time windows. *Expert Systems with Applications*, *38*(1), 435–441. https://doi.org/10.1016/j.eswa.2010.06.082

Zawisza, J. (2018). *Entwicklung und Integration interdependenter Agentensysteme zur dezentralen Produktionsplanung und -steuerung* (Dissertation).

Zenker, M., Emde, S., & Boysen, N. (2016). Cyclic inventory routing in a line-shaped network. *European Journal of Operational Research*, *250*(1), 164–178. https://doi.org/10.1016/j.ejor.2015.10.067

Appendix

Appendix A Werk150

This chapter contains pictures of the real Werk150 at Reutlingen University, the city scooter of the model ‘FlexBlue’ as well as the means of transport considered in this research study.

A.1 Production environment of the Werk150

Appendix 1 shows the production environment of the Werk150 at Reutlingen University.

Appendix 1: Production environment of the Werk150

A.2 Product structure of a city scooter of the model ‘FlexBlue’

The city scooter consists of two main parts, the ‘stem’ and the ‘base’. Appendix 2 shows the individual parts of the stem, Appendix 3 of the base. In the final assembly step, both parts are assembled to a finished city scooter. The finished city scooter can be seen in Appendix 4.

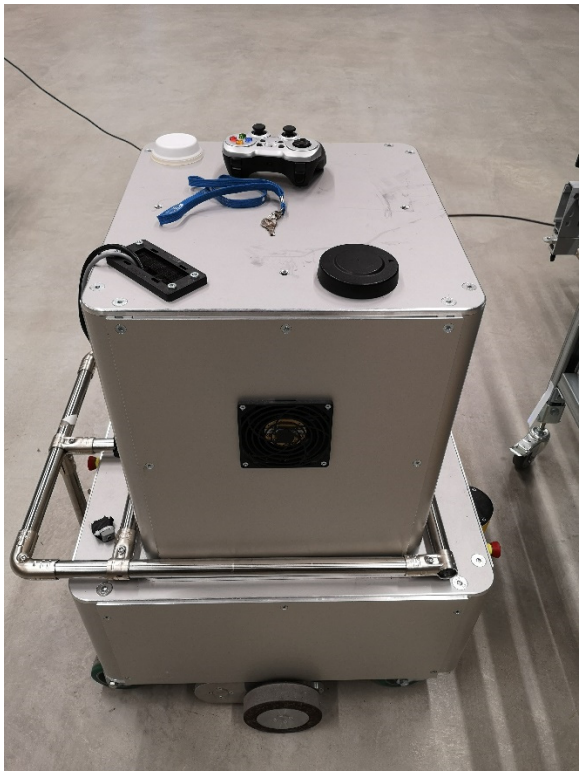
Appendix 2:10 Individual parts of the ‘stem’ of a city scooter

Appendix 3: Individual parts of the ‘base’ of a city scooter

Appendix 4: Complete city scooter of the model 'FlexBlue'

A.3 AGV, NeoKu, KollRo, human

The considered means of transport in this thesis are presented in this section. Pictures of the AGV, NeoKu, KollRo and the human with handcart and tablet are shown. Appendix 5 shows an AGV with a Neobotix platform. The NeoKu with Neobotix platform is shown in Appendix 6. The KollRo, a collaborative tugger train, is presented in Appendix 7. The last means of transport, the handcart with tablet of the human is shown in Appendix 8.



Appendix 5: AGV with Neobotix platform at the Werk150

Appendix 6: NeoKu with Neobotix platform at the Werk150

Appendix 7: KollRo (collaborative tugger train) with waggon at the Werk150

Appendix 8: Handcart with tablet at the Werk150

Appendix B ‘Hettinger table’ to determine the property load handling for the human

In order to determine the property load handling for the human, the Hettinger table is used, which contains information on who is suitable to lift which loads (Hettinger, 1982, p. 96). Appendix 9 shows, with regard to the load handling ordinance, which weights should be used in relation to age and gender at the workplace.

Only guide values for the permissible load weights can be mentioned. The reasonable burden depends in each case on the technical and organisational design of the workplace as well as on the person himself. ‘Occasional’ transport refers to when transportation routes of about three to four steps are carried out less than twice an hour.

Age (years)	Reasonable load for frequency of lifting and carrying			
	Occasionally (up to 2x / h)		More frequent (more than 2x / h)	
	Women	Men	Women	Men
15-18	15 kg	35 kg	10 kg	20 kg
19-45	15 kg	55 kg	10 kg	30 kg
from 45	15 kg	45 kg	10 kg	25 kg
Legend				
	Limits that are recommended from an ergonomic point of view			
	Limit values which are normally not exceeded			

*Appendix 9: Load handling based on(Hettinger, 1982, p. 96)*For the sake of simplification, only the extreme values are considered. Therefore, the human should not carry weights of more than 10 kg.

Appendix C Source code of the algorithm for sequence plan optimisation

In the following, subsections of the source codes of the used operators in the algorithm for sequence plan optimisation are shown.

C.1 Generating an initial population

This source code shows a section of generating an initial population. A description of the source code is given in Section 6.4.2. The source code is shown in Appendix 10.

```
0   int numberOrders;
1   //Initialize.Create population randomly
2   numberOrders = storeOrderInfo.size()/9;//array of transport order
3   for (int i=0; i<numberOrders; i++){
4          random = Math.random();//random number for selecting means of transport
5          if(storeOrderInfo.get(0) < 5 && storeOrderInfo.get(1) > 1){//if transport order is customer order
6                 if (random<0.33){//AGV 1 is selected means of transport
7                        for (int r=0; r<9; r++){
8                               popA1_1.add(storeOrderInfo.get(r));//add information of transport order to means
9   of transport
10                       }
11                       for (int q=0; q<9; q++){
12                              storeOrderInfo.remove(0);//remove selected transport order from array
13                       }
14                }else if (random <0.66){//AGV 2 is selected means of transport
15                       for (int r=0; r<9; r++){
16                              popAgv2_1.add(storeOrderInfo.get(r));//add  information  of  transport  order  to
17  means of transport
18                       }
19                       for (int q=0; q<9; q++){
20                              storeOrderInfo.remove(0);//remove selected transport order from array
21                       }
22                }else{//Human is selected means of transport
23                       for (int r=0; r<9; r++){
24                              popWorker1_1.add(storeOrderInfo.get(r));//add information of transport order to
25  means of transport
26                       }
27                       for (int q=0; q<9; q++){
28                              storeOrderInfo.remove(0);//remove selected transport order from array
29                       }
30                }
31         }else{//if transport order is A part
32                if(random<0.25){//AGV 1 is selected means of transport
33                       for (int r=0; r<9; r++){
34                              popA1_1.add(storeOrderInfo.get(r));//add information of transport order to means
35  of transport
36                       }
37                       for (int q=0; q<9; q++){
38                              storeOrderInfo.remove(0);//remove selected transport order from array
39                       }
40                }else if (random<0.5){//AGV 2 is selected means of transport
41                       for (int r=0; r<9; r++){
42                              popAgv2_1.add(storeOrderInfo.get(r));//add  information  of  transport  order  to
43  means of transport
44                       }
45                       for (int q=0; q<9; q++){
46                              storeOrderInfo.remove(0);//remove selected transport order from array
47                       }
48                }else if (random<0.75){//Human is selected means of transport
49                       for (int r=0; r<9; r++){
50                              popWorker1_1.add(storeOrderInfo.get(r));//add information of transport order to
51  means of transport
52                       }
53                       for (int q=0; q<9; q++){
54                              storeOrderInfo.remove(0);//remove selected transport order from array
55                       }
56                }else{//NeoKu is selected means of transport
57                       for (int r=0; r<9; r++){
58
```

```
59                              popWorker2_1.add(storeOrderInfo.get(r));//add information of transport order to
60  means of transport
61                          }
62                          for (int q=0; q<9; q++){
63                                  storeOrderInfo.remove(0);//remove selected transport order from array
64                          }
65                  }
66          }
67  }
```

Appendix 10: Section of the source code of generating an initial population

C.2 Calculation of the fitness function

This source code shows a section of selecting a chromosome based on its fitness value. The source code shows exemplary the calculation of the fitness value of AGV 1. A description of the source code is given in Section 6.4.3. The source code is shown in Appendix 11.

```
0   //selection. evaluate fitness of each chromosome of the population
1   //calculate transportation time for each population
2   x_positionAgv1 = get_Main().agvs.get(0).getX();//get x position of AGV 1
3   y_positionAgv1 = get_Main().agvs.get(0).getY();//get y position of AGV 1
4   double calcTransportationTimeAgv1_1; //calculation of transportation time of transport order
5   if (popA1_1.isEmpty()==false){
6           calcTransportationTimeAgv1_1                                                                    =
7   calcTransportationTime(popA1_1.get(3),popA1_1.get(4),main.speedAgv,x_positionAgv1,y_positionAgv1);
8           transportationTimeAgv1_1 = calcTransportationTimeAgv1_1+timeFailureAgv1; //in case of a breakdown of
9   the AGV 1, add time for repair
10  }
11  //function of transportation time
12  //get the right pick up and drop off positions of the machines
13  double transportToProduct;
14  double transportToMachine;
15  int x_positionMachineActual;
16  int y_positionMachineActual;
17  int x_positionMachineDesired;
18  int y_positionMachineDesired;
19  if(positionActual == 0){
20          x_positionMachineActual = 170; //x position of goods receipt
21          y_positionMachineActual = 430; //y position of goods receipt
22  }else if(positionActual == 1){
23          x_positionMachineActual = 480; //x position of workstation WS 1.1
24          y_positionMachineActual = 240; //y position of workstation WS 2.1
25  }else if(positionActual == 2){
26          x_positionMachineActual = 520; //x position of workstation WS 1.2
27          y_positionMachineActual = 150; //y position of workstation WS 1.2
28  }else if(positionActual == 3){
29          x_positionMachineActual = 530; //x position of workstation WS 2.1
30          y_positionMachineActual = 240; //y position of workstation WS 2.1
31  }else if(positionActual == 4){
32          x_positionMachineActual = 570; //x position of workstation WS 2.2
33          y_positionMachineActual = 150; //y position of workstation WS 2.2
34  }else if(positionActual == 5){
35          x_positionMachineActual = 575; //x position of workstation wedding / packaging 1
36          y_positionMachineActual = 240; //y position of workstation wedding / packaging 2
37  }else if(positionActual == 6){
38          x_positionMachineActual = 630; //x position of workstation wedding / packaging 2
39          y_positionMachineActual = 150; //y position of workstation wedding / packaging 2
40  }else if(positionActual == 11){
41          x_positionMachineActual = 380; //x position of workstation picking order 1
42          y_positionMachineActual = 180; //y position of workstation picking order 1
43  }else if(positionActual == 12){
44          x_positionMachineActual = 380; //x position of workstation picking order 2
45          y_positionMachineActual = 260; v
46  }else{
47          x_positionMachineActual = storeOrderInfoFailure.get(0); //x position of AGV 1 breakdown
48          y_positionMachineActual = storeOrderInfoFailure.get(1); //y position of AGV 1 breakdown
49  }
50  if(positionDesired == 1){
51          x_positionMachineDesired = 480; //x position of workstation WS 1.1
52          y_positionMachineDesired = 240; //y position of workstation WS 2.1
53  }else if(positionDesired == 2){
54          x_positionMachineDesired = 520; //x position of workstation WS 1.2
55          y_positionMachineDesired = 150; //y position of workstation WS 1.2
56  }else if(positionDesired == 3){
57          x_positionMachineDesired = 530; //x position of workstation WS 2.1
58          y_positionMachineDesired = 240; //y position of workstation WS 2.1
59  }else if(positionDesired == 4){
```

```
60          x_positionMachineDesired = 570; //x position of workstation WS 2.2
61          y_positionMachineDesired = 150; //y position of workstation WS 2.2
62      }else if(positionDesired == 5){
63          x_positionMachineDesired = 575; //x position of workstation wedding / packaging 1
64          y_positionMachineDesired = 240; //y position of workstation wedding / packaging 2
65      }else if(positionDesired == 6){
66          x_positionMachineDesired = 630; //x position of workstation wedding / packaging 2
67          y_positionMachineDesired = 150; //y position of workstation wedding / packaging 2
68      }else if(positionDesired == 11){
69          x_positionMachineDesired = 290; //x position of workstation picking order 1
70          y_positionMachineDesired = 160; //y position of workstation picking order 1
71      }else if(positionDesired == 12){
72          x_positionMachineDesired = 310; //x position of workstation picking order 2
73          y_positionMachineDesired = 330; //y position of workstation picking order 2
74      }else{
75          x_positionMachineDesired = 670; //x position of goods issue
76          y_positionMachineDesired = 470; //y position of goods issue
77      }
78      transportToProduct   =   (Math.sqrt((Math.pow(x_position-x_positionMachineActual,2)   +   Math.pow(y_position-
79      y_positionMachineActual,2))))/speed; //calculate time to move to transport order
80      transportToMachine      =      (Math.sqrt((Math.pow(x_positionMachineActual-x_positionMachineDesired,2)      +
81      Math.pow(y_positionMachineActual-y_positionMachineDesired,2))))/speed; //calculate time to move to the desired
82      destination
83      return transportToProduct + transportToMachine;
84      //end of function transport time
85      //calculate transporttation start of each means of transport
86      transportationStartAgv1                                                                                    =
87      calcTransportationStart(get_Main().agvs.get(0).timeEnd,get_Main().agvs.get(0).transportationTime);
88      //calculate earliest start time of the means of transport
89      //function of transportation time start
90      double newTransportationTime;
91      if(transportationStart+transportationTime < time()){
92          newTransportationTime = time();//means of transport is not in use
93      }else{
94          newTransportationTime = transportationStart+transportationTime; //means of transport is transporting
95      a transport order
96      }
97      return newTransportationTime;
98      // end of function transportation time start
99      //calculate fitness of each population
100     double calcFitnessAgv1_1;
101     if (popA1_1.isEmpty()==false){
102         calcFitnessAgv1_1                                                                                      =
103     calcFitness(popA1_1.get(8),popA1_1.get(6),transportationStartAgv1,transportationTimeAgv1_1);
104         fitnessAgv1_1 = calcFitnessAgv1_1 + transportationTimeAgv1_1;
105     }
106     //function calc fitness
107     double orderDelivery;
108     double transportation;
109     double deliveryReliability;
110     orderDelivery = transportOrderTimeStart + transportOrderDeliveryDate;
111     transportation = transportationStart + transportationTime;
112     deliveryReliability = Math.abs(orderDelivery - transportation); //calculate deviation time of order
113     return deliveryReliability;
114     //end of function calc fitness
115     // normalize fitness value
116     totalFitnessPopulation                                                                                     =
117     fitnessPopulation1+fitnessPopulation2+fitnessPopulation3+fitnessPopulation4+fitnessPopulation5;
118     if (totalFitnessPopulation == infinity){ //in case no transport order is available
119         normalizedfitnessPopulation1 = 0.2;
120     }else{
121         normalizedfitnessPopulation1 = fitnessPopulation1/totalFitnessPopulation; //calculate the probability
122     of getting chosen for mutation or crossover operator
123     }
```

Appendix 11: Section of the source code for calculating the fitness value

C.3 Mutation operator

This source code shows a section of the mutation of a chromosome. The source code shows by way of example the mutation of the first chromosome of a population, mutating the sequence of the transport order of AGV 1. A description of the source code is given in Section 6.4.5.1. The source code is shown in Appendix 12.

```
0     // mutate chromosome with good fitness value
1     if (totalFitnessPopulation != infinity){
2           random = Math.random();//for randomly choosing chromosome
3           random2 = Math.random();//for randomly choosing
4           if (1-normalizedfitnessPopulation1<random){ //first chromosome is chosen if true, add order sequence
5     to new array
6                   for (int i = 0; i<popA1_1.size(); i++){
7                         copyPopulationAgv1.add(popA1_1.get(i));
8                   }
9                   for (int i = 0; i<popAgv2_1.size(); i++){
10                        copyPopulationAgv2.add(popAgv2_1.get(i));
11                  }
12                  for (int i = 0; i<popWorker1_1.size(); i++){
13                        copyPopulationWorker1.add(popWorker1_1.get(i));
14                  }
15                  for (int i = 0; i<popWorker2_1.size(); i++){
16                        copyPopulationWorker2.add(popWorker2_1.get(i));
17                  }
18                  if(popA1_1.isEmpty()==false){ // randomly choosing the line for mutation
19                        getSize = popA1_1.size()/9;
20                        getSize2 = 1/getSize;
21                        n = -1;
22                        for(double i=0; i<random; i=i+getSize2){
23                              n = n+1;
24                        }
25                        for (int i=0; i<9; i++){
26                              mutatedpopAgv2_1.add(popA1_1.get(i+9*n));
27                        }
28                        if (popA1_1.get(0)<5){ //if order is city scooter
29                              if (random2 < 0.5){ //randomly choosing the mutation partner
30                                    for (int i=0; i<9; i++){
31                                          popAgv2_1.add(mutatedpopAgv2_1.get(i));
32                                    }
33                                    for (int i=0; i<9; i++){
34                                          mutatedpopAgv2_1.remove(0);
35                                    }
36                              }else{
37                                    for (int i=0; i<9; i++){
38                                          popWorker1_1.add(mutatedpopAgv2_1.get(i));
39                                    }
40                                    for (int i=0; i<9; i++){
41                                          mutatedpopAgv2_1.remove(0);
42                                    }
43                              }
44                        }else{//if order is A part
45                              if (random2 < 0.33){ //randomly choosing the mutation partner
46                                    for (int i=0; i<9; i++){
47                                          popAgv2_1.add(mutatedpopAgv2_1.get(i));
48                                    }
49                                    for (int i=0; i<9; i++){
50                                          mutatedpopAgv2_1.remove(0);
51                                    }
52                              }else if(random2 < 0.66){
53                                    for (int i=0; i<9; i++){
54                                          popWorker1_1.add(mutatedpopAgv2_1.get(i));
55                                    }
56                                    for (int i=0; i<9; i++){
57                                          mutatedpopAgv2_1.remove(0);
58                                    }
59                              }else{
60                                    for (int i=0; i<9; i++){
61                                          popWorker2_1.add(mutatedpopAgv2_1.get(i));
62                                    }
63                                    for (int i=0; i<9; i++){
64                                          mutatedpopAgv2_1.remove(0);
65                                    }
66                              }
67                        }
68                        for (int i=0; i<9; i++){
69                              popA1_1.remove(0+9*n);
```

```
                        }
                    }
                }
            }
    ///////
            double minFitness;
            minFitness = Math.min(fitnessPopulation1, Math.min(fitnessPopulation2, Math.min(fitnessPopulation3,
    fitnessPopulation4))); //find chromosome with lowest fitness
            if (fitnessPopulation1 == minFitness){ //delete chromosome with lowest fitness and add new solution
                    popA1_1.clear();
                    popAgv2_1.clear();
                    popWorker1_1.clear();
                    popWorker2_1.clear();
                    for (int i = 0; i<copyPopulationAgv1.size(); i++){
                            popA1_1.add(copyPopulationAgv1.get(i));
                    }
                    for (int i = 0; i<copyPopulationAgv2.size(); i++){
                            popAgv2_1.add(copyPopulationAgv2.get(i));
                    }
                    for (int i = 0; i<copyPopulationWorker1.size(); i++){
                            popWorker1_1.add(copyPopulationWorker1.get(i));
                    }
                    for (int i = 0; i<copyPopulationWorker2.size(); i++){
                            popWorker2_1.add(copyPopulationWorker2.get(i));
                    }
            }else if (fitnessPopulation2 == minFitness){ //delete chromosome with lowest fitness and add new
    solution
                    popA1_2.clear();
                    popAgv2_2.clear();
                    popWorker1_2.clear();
                    popWorker2_2.clear();
                    for (int i = 0; i<copyPopulationAgv1.size(); i++){
                            popA1_2.add(copyPopulationAgv1.get(i));
                    }
                    for (int i = 0; i<copyPopulationAgv2.size(); i++){
                            popAgv2_2.add(copyPopulationAgv2.get(i));
                    }
                    for (int i = 0; i<copyPopulationWorker1.size(); i++){
                            popWorker1_2.add(copyPopulationWorker1.get(i));
                    }
                    for (int i = 0; i<copyPopulationWorker2.size(); i++){
                            popWorker2_2.add(copyPopulationWorker2.get(i));
                    }
            }else if (fitnessPopulation3 == minFitness){ //delete chromosome with lowest fitness and add new
    solution
                    popA1_3.clear();
                    popAgv2_3.clear();
                    popWorker1_3.clear();
                    popWorker2_3.clear();
                    for (int i = 0; i<copyPopulationAgv1.size(); i++){
                            popA1_3.add(copyPopulationAgv1.get(i));
                    }
                    for (int i = 0; i<copyPopulationAgv2.size(); i++){
                            popAgv2_3.add(copyPopulationAgv2.get(i));
                    }
                    for (int i = 0; i<copyPopulationWorker1.size(); i++){
                            popWorker1_3.add(copyPopulationWorker1.get(i));
                    }
                    for (int i = 0; i<copyPopulationWorker2.size(); i++){
                            popWorker2_3.add(copyPopulationWorker2.get(i));
                    }
            }else if (fitnessPopulation4 == minFitness){ //delete chromosome with lowest fitness and add new
    solution
                    popA1_4.clear();
                    popAgv2_4.clear();
                    popWorker1_4.clear();
                    popWorker2_4.clear();
                    for (int i = 0; i<copyPopulationAgv1.size(); i++){
                            popA1_4.add(copyPopulationAgv1.get(i));
                    }
                    for (int i = 0; i<copyPopulationAgv2.size(); i++){
                            popAgv2_4.add(copyPopulationAgv2.get(i));
                    }
                    for (int i = 0; i<copyPopulationWorker1.size(); i++){
                            popWorker1_4.add(copyPopulationWorker1.get(i));
                    }
                    for (int i = 0; i<copyPopulationWorker2.size(); i++){
                            popWorker2_4.add(copyPopulationWorker2.get(i));
                    }
            }else{//delete chromosome with lowest fitness and add new solution
                    popA1_5.clear();
                    popAgv2_5.clear();
                    popWorker1_5.clear();
                    popWorker2_5.clear();
                    for (int i = 0; i<copyPopulationAgv1.size(); i++){
```

```
155                         popA1_5.add(copyPopulationAgv1.get(i));
156                 }
157                 for (int i = 0; i<copyPopulationAgv2.size(); i++){
158                         popAgv2_5.add(copyPopulationAgv2.get(i));
159                 }
160                 for (int i = 0; i<copyPopulationWorker1.size(); i++){
161                         popWorker1_5.add(copyPopulationWorker1.get(i));
162                 }
163                 for (int i = 0; i<copyPopulationWorker2.size(); i++){
164                         popWorker2_5.add(copyPopulationWorker2.get(i));
165                 }
166         }
167         copyPopulationAgv1.clear();
168         copyPopulationAgv2.clear();
169         copyPopulationWorker1.clear();
170         copyPopulationWorker2.clear();
171 }
```

Appendix 12: Section of the source code for the mutation operator

C.4 Crossover operator

This source code shows a section of the crossover operation. The source code shows exemplary the crossover of the first chromosome of a population, mutating the sequence of the transport order of AGV 1. A description of the source code is given in Section 6.4.5.2. The source code is shown in Appendix 13.

```
0     // choose crossover partner
1     if (totalFitnessPopulation != infinity){ //if not transport order is available true
2             random2 = Math.random();//create random number
3             if (1-normalizedfitnessPopulation1<random){ //chromosome 1 was chosen
4                     if (random2<0.25){ //randomly select crossover partner based on fitness value
5                             if (1-normalizedfitnessPopulation2<random2){
6                                     repairFunction(popA1_2, mutatedPopulationAgv1, popA1_1); //do crossover and repair new
7     chromosome
8                             }else if (1-(normalizedfitnessPopulation2+normalizedfitnessPopulation3)<random2){
9                                     repairFunction(popA1_3, mutatedPopulationAgv1, popA1_1); //do crossover and repair new
10    chromosome
11                            }else                                                   if                                                  (1-
12    (normalizedfitnessPopulation2+normalizedfitnessPopulation3+normalizedfitnessPopulation4)<random2){
13                                    repairFunction(popA1_4, mutatedPopulationAgv1, popA1_1); //do crossover and repair new
14    chromosome
15                            }else{
16                                    repairFunction(popA1_5, mutatedPopulationAgv1, popA1_1); //do crossover and repair new
17    chromosome
18                            }
19                            crossoverFunction(popAgv2_1, mutatedPopulationAgv1);
20                            crossoverFunction(popWorker1_1, mutatedPopulationAgv1);
21                            crossoverFunction(popWorker2_1, mutatedPopulationAgv1);
22                            popA1_1.clear();
23                            for (int i = 0; i<mutatedPopulationAgv1.size(); i++){
24                                    popA1_1.add(mutatedPopulationAgv1.get(i));
25                            }
26                    }else if (random2<0.5){ //randomly select crossover partner based on fitness value
27
28                            if (1-(normalizedfitnessPopulation2+normalizedfitnessPopulation3)<random2){
29                                    repairFunction(popAgv2_2, mutatedPopulationAgv2, popAgv2_1); //do crossover and repair
30    new chromosome
31                            }else if (1-(normalizedfitnessPopulation2+normalizedfitnessPopulation3)<random2){
32                                    repairFunction(popAgv2_3, mutatedPopulationAgv2, popAgv2_1); //do crossover and repair
33    new chromosome
34                            }else                                                   if                                                  (1-
35    (normalizedfitnessPopulation2+normalizedfitnessPopulation3+normalizedfitnessPopulation4)<random2){
36                                    repairFunction(popAgv2_4, mutatedPopulationAgv2, popAgv2_1); //do crossover and repair
37    new chromosome
38                            }else{
39                                    repairFunction(popAgv2_5, mutatedPopulationAgv2, popAgv2_1); //do crossover and repair
40    new chromosome
41                            }
42
43                            crossoverFunction(popA1_1, mutatedPopulationAgv2);
44                            crossoverFunction(popWorker1_1, mutatedPopulationAgv2);
45                            crossoverFunction(popWorker2_1, mutatedPopulationAgv2);
46                            popAgv2_1.clear();
47                            for (int i = 0; i<mutatedPopulationAgv2.size(); i++){
48                                    popAgv2_1.add(mutatedPopulationAgv2.get(i));
```

```
                        }
                }else if (random2<0.75){ //randomly select crossover partner based on fitness value

                        if                                                                                      (1-
(normalizedfitnessPopulation2+normalizedfitnessPopulation3+normalizedfitnessPopulation4)<random2){
                                repairFunction(popWorker1_2, mutatedPopulationWorker1, popWorker1_1); //do crossover
and repair new chromosome
                        }else if (1-(normalizedfitnessPopulation2+normalizedfitnessPopulation3)<random2){
                                repairFunction(popWorker1_3, mutatedPopulationWorker1, popWorker1_1); //do crossover
and repair new chromosome
                        }else                                           if                                      (1-
(normalizedfitnessPopulation2+normalizedfitnessPopulation3+normalizedfitnessPopulation4)<random2){
                                repairFunction(popWorker1_4, mutatedPopulationWorker1, popWorker1_1);//do crossover
and repair new chromosome
                        }else{
                                repairFunction(popWorker1_5, mutatedPopulationWorker1, popWorker1_1);//do crossover
and repair new chromosome
                        }

                        crossoverFunction(popA1_1, mutatedPopulationWorker1);
                        crossoverFunction(popAgv2_1, mutatedPopulationWorker1);
                        crossoverFunction(popWorker2_1, mutatedPopulationWorker1);
                        popWorker1_1.clear();
                        for (int i = 0; i<mutatedPopulationWorker1.size(); i++){
                                popWorker1_1.add(mutatedPopulationWorker1.get(i));
                        }
                }else{//randomly select crossover partner based on fitness value
                        if                                                                                      (1-
(normalizedfitnessPopulation2+normalizedfitnessPopulation3+normalizedfitnessPopulation4+normalizedfitnessPopulation
5)<random2){
                                repairFunction(popWorker2_2, mutatedPopulationWorker2, popWorker2_1); //do crossover
and repair new chromosome
                        }else if (1-(normalizedfitnessPopulation2+normalizedfitnessPopulation3)<random2){
                                repairFunction(popWorker2_3, mutatedPopulationWorker2, popWorker2_1); //do crossover
and repair new chromosome
                        }else                                           if                                      (1-
(normalizedfitnessPopulation2+normalizedfitnessPopulation3+normalizedfitnessPopulation4)<random2){
                                repairFunction(popWorker2_4, mutatedPopulationWorker2, popWorker2_1); //do crossover
and repair new chromosome
                        }else{
                                repairFunction(popWorker2_5, mutatedPopulationWorker2, popWorker2_1); //do crossover
and repair new chromosome
                        }
                        crossoverFunction(popA1_1, mutatedPopulationWorker2);
                        crossoverFunction(popAgv2_1, mutatedPopulationWorker2);
                        crossoverFunction(popWorker1_1, mutatedPopulationWorker2);
                        popWorker2_1.clear();
                        for (int i = 0; i<mutatedPopulationWorker2.size(); i++){
                                popWorker2_1.add(mutatedPopulationWorker2.get(i));
                        }
                }
        }
        mutatedPopulationAgv1.clear();
        mutatedPopulationAgv2.clear();
        mutatedPopulationWorker1.clear();
        mutatedPopulationWorker2.clear();
}
//crossover function
for (int i = 0; i<collection.size()/9; i++){
        for (int n = 0; n<collectionMutation.size()/9; n++){
                if (collection.isEmpty()==false){
                        if (collection.get(7+9*i)==collectionMutation.get(7+9*n)){ //if collection has same transport
order as mutated collection, remove transport order from collection
                                for (int k = 0; k<9; k++){
                                        collection.remove(0+9*i);
                                }
                                i = 0;
                                n = 0;
                        }
                }
        }
}
//end of crossover function
// repair function
int i = 0;
int compare = 0;
for (i = 0; i<collection_old.size(); i++){
        collectionMutation.add(collection_old.get(i));
}
for (i = 0; i<collection.size()/9; i++){
        if (collectionMutation.isEmpty()==true){ //add transport orders of collection to mutated collection if mutated
collection is empty
                for (int k = 0; k<9; k++){
                        collectionMutation.add(collection.get(k));
                }
```

```
134         }else{
135                 for ( n= 0; n<collectionMutation.size()/9; n++){
136                         if  (collection.get(7+9*i)==collectionMutation.get(7+9*n)){  //check  if  transport  order  is
137   missing
138                                 compare = compare+1;
139                         }
140                 }
141                 if (compare < 1){
142                         for (int k = 0; k<9; k++){
143                                 collectionMutation.add(collection.get(k+9*i));
144                         }
145                 }
146         }
147   }
148   //end of repair function
```

Appendix 13: Section of the source code of the crossover operator

C.5 Mutation operator based on local knowledge

This source code shows a section of mutating a chromosome based on local knowledge. The source code shows exemplary the mutation of the first chromosome of a population, mutating the sequence of the transport order of AGV 1. A description of the source code is given in Section 6.4.5.3. The source code is shown in Appendix 14.

```
0       if (totalFitnessPopulation != infinity){ //calculate shortest utilization time
1               utilizationAgv1 = get_Main().agvs.get(0).utilizationTimePercentage;
2               utilizationAgv2 = get_Main().agvs.get(1).utilizationTimePercentage;
3               utilizationWorker1 = get_Main().workers.get(0).utilizationTimePercentage;
4               utilizationWorker2 = get_Main().kukaIwas.get(0).utilizationTimePercentage;
5               minUtilization    =    Math.min(utilizationAgv1,    Math.min(utilizationAgv2,    Math.min(utilizationWorker1,
6       utilizationWorker2)));
7               double maxFitness;
8               maxFitness    =    Math.max(fitnessPopulation1,    Math.max(fitnessPopulation2,    Math.max(fitnessPopulation3,
9       fitnessPopulation4))); //check chromosome with highest fitness
10              if (fitnessPopulation1 == maxFitness){ //first chromosome has highest fitness
11                      if (popA1_1.size() != 0 && get_Main().agvs.get(0).storeOrderInfoAgv.isEmpty()==true){ //select means
12      of transport with lowest utilization time and send order sequence
13                              if (utilizationAgv1 == minUtilization){
14                                      for (int i = 0; i<9; i++){
15                              get_Main().agvs.get(0).populationAgv1.add(popA1_1.get(i));
16                                      }
17                              }else if (utilizationAgv2 == minUtilization){
18                                      for (int i = 0; i<9; i++){
19                              get_Main().agvs.get(1).populationAgv1.add(popA1_1.get(i));
20                                      }
21                              }else if(utilizationWorker1 == minUtilization){
22                                      for (int i = 0; i<9; i++){
23                              get_Main().workers.get(0).populationAgv1.add(popA1_1.get(i));
24                                      }
25                              }else{
26                                      for (int i = 0; i<9; i++){
27                              get_Main().kukaIwas.get(0).populationAgv1.add(popA1_1.get(i));
28                                      }
29                              }
30                              for (int i = 0; i<9; i++){
31                                      populationAgv1.add(popA1_1.get(i));
32                              }
33                              for (int i = 0; i<9; i++){
34                                      popA1_1.remove(0);
35                              }
36                              if (popA1_1.isEmpty() == false){
37                                      for (int i = 0; i<popA1_1.size(); i++){
38                                              storeOrderInfo.add(popA1_1.get(i));
39                                              storeOrderInfo1.add(popA1_1.get(i));
40                                              storeOrderInfo2.add(popA1_1.get(i));
41                                              storeOrderInfo3.add(popA1_1.get(i));
42                                              storeOrderInfo4.add(popA1_1.get(i));
43                                      }
44                              }
45                      }else{
46                              for (int i = 0; i<popA1_1.size(); i++){
47                                      storeOrderInfo.add(popA1_1.get(i));
48                                      storeOrderInfo1.add(popA1_1.get(i));
49                                      storeOrderInfo2.add(popA1_1.get(i));
50                                      storeOrderInfo3.add(popA1_1.get(i));
```

```
51                                  storeOrderInfo4.add(popA1_1.get(i));
52                          }
53                  }
54          if (utilizationAgv1 == minUtilization){ //send sequence of transport orders to AGV 1
55                  send("newOrder", get_Main().agvs.get(0));
56          }
57  }
58  popA1_1.clear();
59  if (idMachine == 0){ //AGV 1 modifies the order sequence
60          if(populationWorker1.isEmpty()==false){
61                  for (int i = 0; i<9; i++){
62                          populationAgv1.add(populationWorker1.get(i));
63                  }
64                  for (int i = 0; i<9; i++){
65                          populationWorker1.remove(0);
66                  }
67          }else if(populationAgv2.isEmpty()==false){
68                  for (int i = 0; i<9; i++){
69                          populationAgv1.add(populationAgv2.get(i));
70                  }
71                  for (int i = 0; i<9; i++){
72                          populationAgv2.remove(0);
73                  }
74          }else{
75                  if (populationWorker2.isEmpty()==false){
76                          for (int i = 0; i<9; i++){
77                                  populationAgv1.add(populationWorker2.get(i));
78                          }
79                          for (int i = 0; i<9; i++){
80                                  populationWorker2.remove(0);
81                          }
82                  }
83          }
84          for (int i = 0; i<populationAgv1.size(); i++){
85                  get_Main().manager.proposedPopulationAgv1.add(populationAgv1.get(i));
86          }
87          for (int i = 0; i<populationAgv2.size(); i++){
88                  get_Main().manager.proposedPopulationAgv2.add(populationAgv2.get(i));
89          }
90          for (int i = 0; i<populationWorker1.size(); i++){
91                  get_Main().manager.proposedPopulationWorker1.add(populationWorker1.get(i));
92          }
93          for (int i = 0; i<populationWorker2.size(); i++){
94                  get_Main().manager.proposedPopulationWorker2.add(populationWorker2.get(i));
95          }
96  }else{
97          if(populationWorker1.isEmpty()==false){
98                  for (int i = 0; i<9; i++){
99                          populationAgv2.add(populationWorker1.get(i));
100                 }
101                 for (int i = 0; i<9; i++){
102                         populationWorker1.remove(0);
103                 }
104         }else if(populationAgv1.isEmpty()==false){
105                 for (int i = 0; i<9; i++){
106                         populationAgv2.add(populationAgv1.get(i));
107                 }
108                 for (int i = 0; i<9; i++){
109                         populationAgv1.remove(0);
110                 }
111         }else{
112                 if (populationWorker2.isEmpty()==false){
113                         for (int i = 0; i<9; i++){
114                                 populationAgv2.add(populationWorker2.get(i));
115                         }
116                         for (int i = 0; i<9; i++){
117                                 populationWorker2.remove(0);
118                         }
119                 }
120         }
121         for (int i = 0; i<populationAgv1.size(); i++){
122                 get_Main().manager.proposedPopulationAgv1.add(populationAgv1.get(i));
123         }
124         for (int i = 0; i<populationAgv2.size(); i++){
125                 get_Main().manager.proposedPopulationAgv2.add(populationAgv2.get(i));
126         }
127         for (int i = 0; i<populationWorker1.size(); i++){
128                 get_Main().manager.proposedPopulationWorker1.add(populationWorker1.get(i));
129         }
130         for (int i = 0; i<populationWorker2.size(); i++){
131                 get_Main().manager.proposedPopulationWorker2.add(populationWorker2.get(i));
132         }
133 }
134 send("newProposedOrder",get_Main().manager); //AGV 1 sends order sequence back to manager
135 populationAgv1.clear();
```

```
populationAgv2.clear();
populationWorker1.clear();
populationWorker2.clear();
if (totalFitnessPopulation != infinity){
	//calculate transportation time for old population
	double calcTransportationTimePopulationAgv1;
	transportationTimeAgv1 = 0;
	if (populationAgv1.isEmpty()==false){
		calcTransportationTimePopulationAgv1 =
calcTransportationTime(populationAgv1.get(3),populationAgv1.get(4),main.speedAgv,x_positionAgv1,y_positionAgv1);
		transportationTimeAgv1 = calcTransportationTimePopulationAgv1+timeFailureAgv1;
	}
	//calculate transportation time for proposed population
	double calcTransportationTimeAgv1;
	proposedTransportationTimeAgv1 = 0;
	if (proposedPopulationAgv1.isEmpty()==false){
		calcTransportationTimeAgv1 =
calcTransportationTime(proposedPopulationAgv1.get(3),proposedPopulationAgv1.get(4),main.speedAgv,x_positionAgv1,y_p
ositionAgv1);
		proposedTransportationTimeAgv1 = calcTransportationTimeAgv1+timeFailureAgv1;
	}
double proposedTransportationStartAgv1;
proposedTransportationStartAgv1 =
calcTransportationStart(get_Main().agvs.get(0).timeEnd,get_Main().agvs.get(0).transportationTime);
//calculate fitness of old population
double calcFitnessPopulationAgv1;
double fitnessPopulationAgv1 = 0;
if (populationAgv1.isEmpty()==false){
	calcFitnessPopulationAgv1 =
calcFitness(populationAgv1.get(8),populationAgv1.get(6),transportationStartAgv1,transportationTimeAgv1);
		fitnessPopulationAgv1 = calcFitnessPopulationAgv1 + transportationTimeAgv1;
	}
//calculate fitness of proposed population
double calcFitnessAgv1;
double fitnessAgv1 = 0;
if (proposedPopulationAgv1.isEmpty()==false){
	calcFitnessAgv1 =
calcFitness(proposedPopulationAgv1.get(8),proposedPopulationAgv1.get(6),transportationStartAgv1,proposedTransportat
ionTimeAgv1);
	fitnessAgv1 = calcFitnessAgv1 + proposedTransportationTimeAgv1;
}
//compare proposed population with old population
	if(fitnessPopulation<=proposedFitnessPopulation){
		if (proposedPopulationAgv1.size() != 0){
			for (int i = 0; i<proposedPopulationAgv1.size(); i++){
				get_Main().agvs.get(0).storeOrderInfoAgv.add(proposedPopulationAgv1.get(i));
			}
			get_Main().agvs.get(0).transportationTimeCollection.add(proposedTransportationTimeAgv1);
			get_Main().agvs.get(0).counterTransportOrder =
get_Main().agvs.get(0).counterTransportOrder+1;
			send("needed", get_Main().agvs.get(0));
		}
	}
	populationAgv1.clear();
}
```

Appendix 14: Section of the source code of the mutation operator with local knowledge

Appendix D Choice of the appropriate simulation software

For the implementation of the developed system, a suitable simulation software application was initially searched for. Due to the large quantity of possible software on the market, only the simulation material relevant in the area of production and logistics was considered. Initially, the requirements for such a system were determined for the analysis. The requirements were subsequently ranked by their importance using the pairwise comparison method, and finally evaluated in a benefit analysis.

D.1 Requirements of a simulation software

The analysis of the software programs focuses on the following five categories:

- Programming environment: For successful development of the system, the simulation software should have a simple and intuitive user interface. In addition to existing building block libraries, these should be expandable by a programming language in order to generate as much individualisation as possible.
- Animation & Visualisation: Animation and visualisation are integral parts of the V&V process of the system to be developed. Among other things, the system can be checked for plausibility. In addition, the behaviour of the system can be checked carefully during the programming phase.
- Analysis: In order to check the system for its suitability, the simulation program must be able to carry out various analyses. Here, the simulation program should be able to analyse different values of the system and to compare them with each other. The results, for example, should then be graphically presented in diagrams.
- Data transfer: Since the system is later implemented in the Werk150, the simulation software must offer an interface to external systems. It is important that data can be received and sent to external software programs.
- Support: The last aspect that is taken into account for the choice of software program is the support in programming from the company. It is important that different sources of information are available to respond quickly to possible programming problems.

D.2 Description of the simulation software

The analysis focuses on the following selected simulation software: AnyLogic, Demo3D, Enterprise Dynamics, Plant Simulation and Simul8. The software programs are described below in accordance with the above-mentioned requirements. The gathered information is based on internet research, email and telephone calls and if possible, by testing free trial versions.

AnyLogic: AnyLogic offers the ability to combine all three simulation methods, namely discrete event, agent-based and system dynamics, to simulate models with any complexity. The programming environment contains a building block library which can be used per drag-and-drop. The program offers several modelling languages like process flowchart, state charts, action charts and stock-and-flow diagrams. Additionally, models developed in AnyLogic are fully mapped into Java code and linked with the AnyLogic simulation engine. Users can use animation to validate a dynamic model, allowing them to identify process bottlenecks and highlight results. Animations can be created hierarchically, with multiple 2D or 3D views. AnyLogic provides custom objects and reusable object libraries for the application areas. A set of predefined experiments allows exploring the simulation. Furthermore, it is possible to create customised experiments and create own algorithms and optimisation engines with Java. Importing custom 3D models, images, CAD drawings, and shape files into the simulation is possible. Data storage, including Oracle, MS SQL, MySQL, PostgreSQL, MS Access, Excel, and text files can be integrated in the simulation. Also, the simulation model can be embedded in an operational software like ERP or MRP and shared via the AnyLogic Cloud. AnyLogic offers consultative support, trainings and events. Furthermore, case studies, books, white papers, blogs, academic articles are available and freely accessible. Also, industry-specific libraries can be used for free to support the simulation (AnyLogic, 2019).

Demo3D: The user is able to model its simulation with discrete-event modelling method. The construction of models takes place with the help of parameterisable standard and / or customer-specific blocks from catalogues. The elements are dragged and dropped into the models. The simulation is expandable by using the programming language JScript, or alternatively using the C# for individualising the simulation model. The catalogue concept ensures a high degree of reuse by providing numerous building blocks like QuickStart, people, robots and buildings. These are supplemented by components provided online, in the so-called Webstore. 3D graphics are used for the visual presentation of the simulation model. For the analysis requirements, Demo3D provides a set of pre-designed experiments. Demo3D allows the import

of numerous CAD formats (SolidWorks, Sketchup, Inventor, DXF, DWG, VRML), which further enhances the level of detail of the models. The export of data (VRML, DWG, DXF) is also possible. Support is available through consulting, tutorials available on YouTube, as well as training programmes (Demo3D, 2019).

Enterprise Dynamics: The program environment consists of an object-orientated modelling platform using discrete-event modelling paradigm. Enterprise Dynamics offers branch-specific object libraries. Via drag-and-drop, the objects are placed into the simulation program. If required, the objects can be created and / or modified individually to fit specific needs. Therefore, the modelling languages Delphi 7, C++ or 4DScript can be used. The simulated model can be analysed and visualised in 2D, 3D or VR animation. In addition to the existing reusable object libraries for different applications, it is possible to create own object libraries and modify existing objects. The experimentation wizard offers a set of pre-defined scenarios for the model. With the virtual optimisation test, any scenario can be tested and be improved throughout the entire system lifecycle. Enterprise Dynamics, if needed can be integrated with external data source and third-party systems. Interfaces exist with data bases like SQL, Excel or Access as well as with graphical data e.g. Bitmap, *.wmf, *.dxf, VRML. External connections with OPC-Server, TCP / IP is also possible. Enterprise Dynamics offers training for beginners, advanced users and teachers. Also, free introductory webinars are available besides phone support and video tutorials (Enterprise Dynamics, 2019).

Plant Simulation: Plant Simulation offers an object-orientated modelling environment. The modelling method is discrete-event. The objects are placed in the model with a drag-and-drop function. The freely configurable block library contains all plant simulation basic and user blocks that are visible and accessible. Any user blocks are created graphically and interactively by the user himself from basic blocks. SimTalk is the modelling language used in Plant Simulation. Plant Simulation allows 2D and 3D representation of production processes. In addition, it allows the presentation of complete plant concepts at an early stage in a virtual, interactive, immersive environment. For example, 3D design data can be dragged and dropped from Solid Edge in jt-format. Plant Simulation provides predefined experiments as well as statistical tools like DataFit, Goodness-of-Fit Test, ANOVA, regression analysis or confidence analysis for analysis of the simulation model. The program also offers optimisation tools, e.g. neural networks. Plant Simulation can be connected with different programs like ARIS, ActiveX, CDDE, Mailbox, ODBC (e.g. MS Access), Socket SQL (e.g. ORACLE, MS SQL Server), also during the simulation. The import of CAD data is also possible. In case of needed

support, training courses are offered and YouTube videos are available. In addition, email and phone support exists (Plant Simulation, 2019).

Simul8: The programming environment is object-orientated. The user is able to use existing libraries like value stream mapping, material handling, continuous flow simulation, agent-based simulation or business process modelling notation. Simul8 uses an intuitive drag and drop interface for building simulations. The used programming language is VisualLogic, which allows the user to implement detailed logic of the simulation. The integration of customised 2D or 3D animation is possible. Simul8 provides a built-in graphics library for the application areas. To analysze simulation models Simul8 provides its users with different options like sensitivity analysis, distribution fitting using Stat::Fit, optimising simulations with OptQuest or testing multiple configurations with the scenario manager. It is possible to test real scenarios in a virtual environment, for example, by simulating planned function and load tests of the system, changing parameters affecting system performance, carrying out extreme-load tests or verifying experiments. The design of SIMUL8 also facilitates communication with other software packages such as Microsoft Access, Excel and Visio. The support of XML and OLE automation allows working with external sources of data and exporting internal data to other systems. SIMUL8 also supports communication with databases using SQL. Simul8 supports its users with training classes, online coaching, online help files, consulting and case studies. Additionally, phone support is offered and YouTube tutorial videos are available (Simul8, 2019).

D.3 Results of the benefit analysis

The following benefit analysis relies on the above-mentioned requirements for a simulation program. Prior to the benefit analysis, a pairwise comparison shown in Appendix 15 takes place. With this method, the individual objects are compared in pairs and rated, according to which of each entity is preferred.

In the evaluation matrix, the individual requirements receive different weightings, depending on their significance. For this purpose, two requirements are always compared with each other according to their importance for the undertaken study. If one requirement is more important than the other one, it receives two points and in return, the other requirement receives zero points. If both requirements are equally important, both get one point. The value of each requirement results from the sum of the points per line. Finally, the calculation of the weighting of the requirements follows as a percentage of the total score. According to the evaluation, the

aspects programming environment, analysis and data transfer are particularly crucial for the success of the simulation programming. The percentage weight of the requirements forms the basis for the following benefit analysis.

	Programming environment	Animation & Visualisation	Analysis	Data transfer	Support	Total	Weighting
Programming environment		2	1	2	2	7	35%
Animation & Visualisation	0		1	0	1	2	10%
Analysis	1	1		1	2	5	25%
Data transfer	0	2	1		2	5	25%
Support	0	1	0	0		1	5%

Rating	Description
0	less important
1	equally important
2	more important

Appendix 15: Pairwise comparison of the requirements for the simulation program

Appendix 16 shows the carried out benefit analysis. The evaluation of the respective simulation programs takes place, based on numbers from 0 to 5. The rating number 0 means that the alternative does not meet the requirements. A rating number of 5, on the other hand, means that the solution approach fully meets the requirements.

The simulation program of AnyLogic scored best with 4.05 out of 5 points, taking into account the above-mentioned requirements. The requirements which are of high importance are particularly well met by AnyLogic.

		AnyLogic		Demo3D		Enterprise Dynamics		Plant Simulation		Simul8	
	Weighting	Average of ratings	Value	Average of ratings	Value	Average of ratings	Value	Average of ratings	Value	Average of ratings	Value
Programming environment	35%	5	1.75	2	0.70	2	0.70	3	1.05	4	1.40
Animation & Visualisation	10%	4	0.40	3	0.30	3	0.30	4	0.40	4	0.40
Analysis	25%	3	0.75	3	0.75	4	1.00	5	1.25	3	0.75
Data transfer	25%	4	1.00	3	0.75	3	0.75	4	1.00	3	0.75
Support	5%	3	0.15	2	0.10	5	0.25	4	0.20	4	0.20
	Sum		**4.05**		**2.60**		**3.00**		**3.90**		**3.50**

Rating	Description
1	Bad
2	Not good
3	Good
4	Very good
5	Excellent

Appendix 16: Benefit analysis of the simulation programs

Appendix E Simulation experiment

This chapter contains the required data and calculations for the V&V process of the autonomous system. The calculations for the confidence intervals are provided for this purpose. In addition, the data obtained from the simulation and validation in the Werk150 are presented. Due to the large amount of data only a sample of the obtained data can be shown. In addition, the individual confidence intervals of the logistic target values for the respective simulation run are shown in tables.

E.1 Calculation of the confidence intervals

For calculating a confidence interval a measurement series $x_1, ..., x_n$ with n independent random variables $X_1, \ldots, X_n$ is taken as a starting point. The confidence interval determines an interval in which the expected value of the random variable θ lies with a certain probability. The interval [U; O] is usually determined that the expected value of the random variable is with a probability of 1 - α within the range ($P(U \leq \theta \leq O) = 1 - \alpha$). Accordingly, 1 - α is referred to as confidence probability. In contrast, α is the probability of error with which the determined interval does not contain the expected value. The term for the interval [U; O] is called confidence interval (Gutenschwager et al., 2017, pp. 114–115).

The interval boundaries U and O result from the random variables $X_1, \ldots, X_n$, hence $U = u(X_1, \ldots, X_n)$ and $O = o(X_1, \ldots, X_n)$ with $u: \mathbb{R}^n \rightarrow \mathbb{R}$ and $o: \mathbb{R}^n \rightarrow \mathbb{R}$. For the expected value $\mu = E(x)$ of a random variable X, the confidence interval can be specified as follows (for the derivation, see (Fahrmeir, Heumann, Künstler, Pigeot, & Tutz, 2016, pp. 358–363)):

$$\left[\bar{x} - z_{1-\frac{\alpha}{2}}\frac{s}{\sqrt{n}}; \bar{x} + z_{1-\frac{\alpha}{2}}\frac{s}{\sqrt{n}}\right] \tag{E.1}$$

with:

$\bar{x}$	mean of the measured values
z	standard normal random variable
α	confidence interval
s	empirical standard deviation
n	sample size

Thereby, $\bar{x}$ is the mean value of the measured values x_i of a characteristic X with i = 1, ..., n and n number of observations, which is defined as follows:

$$\bar{x} = \frac{1}{n}(x_i + \cdots + x_n) = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{E.2}$$

with: $\bar{x}$ mean of the measured values
n sample size
x_i value of sample i

The standard deviation s indicates the fluctuation of the values around the mean:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{E.3}$$

with: s empirical standard deviation
n sample size
x_i value of sample i
$\bar{x}$ mean of the measured values

A normal distribution with the values $\mu = 0$ and $\sigma^2 = 1$ (N(0, 1) distribution) is called a standard normal distribution. Each normally distributed random variable X can be traced back to a standard normally distributed random variable Z as follows:

$$Z = \frac{X - \mu}{\sigma} \tag{E.4}$$

with: Z standard normal random variable
X random variable
μ mean
σ standard deviation

E.2 Simulation data of the static transport system

The following data was recorded during the simulation runs for the statistical verification. As input variable for the simulation runs the parameter ‘time until the next order of a new city scooter’ was chosen. The selected input variables are based on steps of 100 seconds and laying in the range from 100 to 1,000 seconds. For each input variable, 100 simulation runs were carried out. The data of the static system are shown in Figure Appendix 17 and Appendix 18, for the autonomous system in Figure Appendix 19 and Appendix 20.

	Input value: 100 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	4008.268	2489.318	2904.139	3078.974	4573.002	...	1755.061	1637.18	**2367.7059**
	ScheduleDeviation [s]	3420.018	1852.485	2236.039	2423.617	3878.86	...	1110.661	986.93	**1715.54427**
	UtilizationTime AGV 1 [%]	0.14310954	0.33710692	0.19723183	0.38599349	0.32307692	...	0.34644195	0.28810409	**30.0477854**
	UtilizationTime AGV 2 [%]	0.48409894	0.36855346	0.31314879	0.29641694	0.46627219	...	0.26779026	0.25092937	**32.7972246**
	UtilizationTime Human [%]	0.24204947	0.20628931	0.30449827	0.22638436	0.13846154	...	0.24344569	0.28252788	**25.8685316**
	UtilizationTime Kukaliwa [%]	0.13074205	0.08805031	0.18512111	0.09120521	0.07218935	...	0.1423221	0.17843866	**11.2864584**
	Ratio of EmptyRuns to Utilization Time [%]	0.35425	0.3675	0.39075	0.3645	0.3465	...	0.37025	0.36125	**35.60275**
Target system KollRo	TransportationTime [s]	65.802	168.197	116.484	88.858	142.639	...	91.306	194.523	**104.44298**
	ScheduleDeviation [s]	10965.064	7928.278	8303.473	8430.076	10490.599	...	9569.358	7002.866	**9061.65693**
	UtilizationTime [%]	0.161	0.12	0.132	0.154	0.132	...	0.133	0.126	**13.631**
	WaitingTime [%]	0.839	0.88	0.868	0.846	0.868	...	0.867	0.874	**86.369**
	EmptyRuns [%]	0.898	0.957	0.869	0.89	0.811	...	0.864	0.823	**89.079**
	Input value: 200 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	1304.667	3114.134	2926.062	1351.26	1305.539	...	1889.194	2088.026	**1870.00168**
	ScheduleDeviation [s]	692.111	2526.194	2334.312	723.331	642.95	...	1208.909	1447.719	**1220.96765**
	UtilizationTime AGV 1 [%]	0.22777778	0.19169329	0.21840874	0.46615385	0.33333333	...	0.24595469	0.32624113	**32.5741873**
	UtilizationTime AGV 2 [%]	0.25277778	0.2971246	0.32605304	0.12	0.33333333	...	0.23624595	0.35258359	**29.3943161**
	UtilizationTime Human [%]	0.37222222	0.36421725	0.24960998	0.32307692	0.28813559	...	0.36893204	0.21884498	**26.4425432**
	UtilizationTime Kukaliwa [%]	0.14722222	0.14696486	0.20592824	0.09076923	0.04519774	...	0.14886731	0.10233029	**11.5889535**
	Ratio of EmptyRuns to Utilization Time [%]	0.297	0.39975	0.35825	0.38325	0.40875	...	0.38975	0.356	**37.1795**
Target system KollRo	TransportationTime [s]	90.863	124.171	144.01	84.61	190.366	...	93.664	111.908	**108.89009**
	ScheduleDeviation [s]	10442.527	8256.42	8756.555	6259.946	6223.864	...	10207.837	10070.882	**8927.32815**
	UtilizationTime [%]	0.11	0.097	0.106	0.135	0.099	...	0.1	0.131	**11.65**
	WaitingTime [%]	0.89	0.903	0.894	0.865	0.901	...	0.9	0.869	**88.35**
	EmptyRuns [%]	0.917	1	0.975	0.929	1	...	1	0.812	**92.286**
	Input value: 300 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	1134.372	1039.872	1841.706	1396.546	1107.396	...	1465.105	1631.865	**1634.34664**
	ScheduleDeviation [s]	475.728	396	1219.625	750.561	498.75	...	861.425	967.896	**997.88788**
	UtilizationTime AGV 1 [%]	0.11764706	0.18581907	0.21114865	0.28095238	0.17532468	...	0.23746702	0.3256262	**25.6339601**
	UtilizationTime AGV 2 [%]	0.10845588	0.21026895	0.38513514	0.18571429	0.19480519	...	0.21372032	0.28131021	**24.276975**
	UtilizationTime Human [%]	0.72426471	0.48410758	0.27871622	0.39285714	0.50974026	...	0.45118734	0.28516378	**39.1944826**
	UtilizationTime Kukaliwa [%]	0.04963235	0.1198044	0.125	0.14047619	0.12012987	...	0.09762533	0.10789981	**10.8945823**
	Ratio of EmptyRuns to Utilization Time [%]	0.37575	0.41325	0.3875	0.40425	0.42525	...	0.377	0.398	**38.5515**
Target system KollRo	TransportationTime [s]	86.809	122.913	89.179	149.421	42.215	...	93.444	205.576	**97.92301**
	ScheduleDeviation [s]	2928.563	10162.555	9244.653	7648.636	9831.607	...	7604.871	7140.081	**7996.83348**
	UtilizationTime [%]	0.153	0.107	0.114	0.103	0.123	...	0.115	0.091	**11.694**
	WaitingTime [%]	0.847	0.893	0.886	0.897	0.877	...	0.885	0.909	**88.306**
	EmptyRuns [%]	0.938	0.955	0.87	0.939	0.943	...	0.929	0.981	**93.894**
	Input value: 400 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	1741.663	800.958	816.048	785.259	766.544	...	769.166	757.746	**1176.70018**
	ScheduleDeviation [s]	1146.947	135.386	198.589	134.451	111.558	...	119.316	121.492	**536.47044**
	UtilizationTime AGV 1 [%]	0.17597293	0.10958904	0.14134276	0.11818182	0.08979592	...	0.4509018	0.19817073	**17.928579**
	UtilizationTime AGV 2 [%]	0.19966159	0.10616438	0.14134276	0.11818182	0.34693878	...	0.0741483	0.07012195	**17.9572552**
	UtilizationTime Human [%]	0.55837563	0.68835616	0.62190813	0.67575758	0.35102041	...	0.43486974	0.60670732	**53.077346**
	UtilizationTime Kukaliwa [%]	0.06598985	0.09589041	0.09540636	0.08787879	0.2122449	...	0.04008016	0.125	**11.0368198**
	Ratio of EmptyRuns to Utilization Time [%]	0.38625	0.41625	0.444	0.43925	0.403	...	0.395	0.4245	**40.1124167**
Target system KollRo	TransportationTime [s]	87.148	89.905	92.756	94.765	90.076	...	44.274	65.277	**101.26217**
	ScheduleDeviation [s]	3828.108	8663.737	8975.305	6650.97	10042.446	...	7582.657	9621.532	**7331.83171**
	UtilizationTime [%]	0.133	0.113	0.111	0.116	0.113	...	0.127	0.12	**11.854**
	WaitingTime [%]	0.867	0.887	0.889	0.884	0.887	...	0.873	0.88	**88.146**
	EmptyRuns [%]	0.952	0.987	0.936	0.984	0.971	...	1	0.952	**92.815**
	Input value: 500 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	750.452	1967.715	745.344	1209.262	1053.66	...	1352.463	743.611	**1061.50029**
	ScheduleDeviation [s]	126.867	1386.132	147.826	568.405	410.928	...	762.876	120.591	**433.09959**
	UtilizationTime AGV 1 [%]	0.16019417	0.19105691	0.46461949	0.17940199	0.17811705	...	0.28096677	0.10625	**19.9227877**
	UtilizationTime AGV 2 [%]	0.33786408	0.12601626	0.22830441	0.1461794	0.17811705	...	0.33232628	0.184375	**17.9991729**
	UtilizationTime Human [%]	0.24854369	0.47560976	0.27102804	0.58471761	0.55979644	...	0.28096677	0.5875	**52.265109**
	UtilizationTime Kukaliwa [%]	0.25339806	0.20731707	0.03604806	0.089701	0.08396947	...	0.10574018	0.121875	**9.81293037**
	Ratio of EmptyRuns to Utilization Time [%]	0.407	0.41275	0.377	0.3905	0.419	...	0.42675	0.42833333	**41.1090833**
Target system KollRo	TransportationTime [s]	91.302	92.8	123.617	158.002	121.204	...	121.636	145.105	**104.0273**
	ScheduleDeviation [s]	6127.168	6221.495	5775.514	7025.92	6871.061	...	8365.353	9865.872	**6604.8202**
	UtilizationTime [%]	0.116	0.107	0.105	0.1	0.107	...	0.1	0.103	**11.58**
	WaitingTime [%]	0.884	0.893	0.895	0.9	0.893	...	0.9	0.897	**88.42**
	EmptyRuns [%]	0.978	0.957	1	1	0.954	...	0.906	0.947	**95.157**

Appendix 17: Simulation runs of the static system for the input values between 100 and 500 seconds

	Input value: 600 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	701.912	1117.415	695.397	705.179	1167.194	...	701.209	699.166	**923.57741**
	ScheduleDeviation [s]	55.974	475.272	91.996	83.117	540.532	...	84.599	110.774	**304.58307**
	UtilizationTime AGV 1 [%]	0.1598513	0.17100372	0.25810811	0.06769231	0.20344828	...	0.24875622	0.20114286	**18.6380564**
	UtilizationTime AGV 2 [%]	0.14126394	0.14498141	0.46756757	0.53384615	0.17931034	...	0.43283582	0.41028571	**21.7348282**
	UtilizationTime Human [%]	0.58736059	0.55018587	0.22972973	0.35076923	0.50344828	...	0.27736318	0.336	**51.3889987**
	UtilizationTime Kukaliwa [%]	0.11152416	0.133829	0.04459459	0.04769231	0.1137931	...	0.04104478	0.05257143	**8.23811674**
	Ratio of EmptyRuns to Utilization Time [%]	0.422	0.4255	0.425	0.43	0.42	...	0.426	0.417	**42.7549167**
Target system KollRo	TransportationTime [s]	67.052	91.136	95.197	43.577	153.508	...	231.06	175.607	**99.12817**
	ScheduleDeviation [s]	7328.909	8036.828	6432.476	6094.869	6496.499	...	5442.076	3829.395	**5831.94945**
	UtilizationTime [%]	0.121	0.11	0.114	0.128	0.101	...	0.09	0.109	**11.649**
	WaitingTime [%]	0.879	0.89	0.886	0.872	0.899	...	0.91	0.891	**88.351**
	EmptyRuns [%]	1	0.982	0.953	0.966	0.975	...	1	0.928	**95.907**
	Input value: 700 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	1173.546	873.979	677.546	1105.015	686.371	...	980.257	695.167	**870.19963**
	ScheduleDeviation [s]	556.468	292.016	100.095	507.179	57.236	...	382.683	89.18	**265.66839**
	UtilizationTime AGV 1 [%]	0.17021277	0.09701493	0.09570957	0.20833333	0.328125	...	0.08823529	0.07943925	**17.4519614**
	UtilizationTime AGV 2 [%]	0.17021277	0.09701493	0.09240924	0.20192308	0.15364583	...	0.16176471	0.12616822	**17.2757238**
	UtilizationTime Human [%]	0.60283688	0.73880597	0.74257426	0.50961538	0.4453125	...	0.61029412	0.69158879	**57.5757318**
	UtilizationTime Kukaliwa [%]	0.05673759	0.06716418	0.06930693	0.08012821	0.07291667	...	0.13970588	0.10280374	**7.69658294**
	Ratio of EmptyRuns to Utilization Time [%]	0.44675	0.42475	0.432	0.443	0.437	...	0.45525	0.427	**42.9878333**
Target system KollRo	TransportationTime [s]	145.699	120.642	153.414	42.602	65.722	...	64.977	91.331	**107.24373**
	ScheduleDeviation [s]	2534.928	4402.343	5143.397	6438.503	7352.71	...	6913.868	8292.43	**5112.60635**
	UtilizationTime [%]	0.114	0.106	0.102	0.128	0.119	...	0.117	0.111	**11.55**
	WaitingTime [%]	0.886	0.894	0.898	0.872	0.881	...	0.883	0.889	**88.45**
	EmptyRuns [%]	1	0.94	0.932	0.942	0.937	...	0.981	0.941	**94.133**
	Input value: 800 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	1243.252	1123.995	678.386	680.458	1274.182	...	672.331	966.131	**846.13165**
	ScheduleDeviation [s]	590.908	515.889	75.787	87.022	656.694	...	70.078	369.911	**239.30959**
	UtilizationTime AGV 1 [%]	0.15408805	0.09854015	0.08405797	0.22446406	0.22175732	...	0.34424603	0.07471264	**18.9786205**
	UtilizationTime AGV 2 [%]	0.17924528	0.11313869	0.2057971	0.45271122	0.18828452	...	0.18650794	0.07758621	**18.3004015**
	UtilizationTime Human [%]	0.61006289	0.73722628	0.66086957	0.28121059	0.49372385	...	0.2172619	0.80172414	**55.0092361**
	UtilizationTime Kukaliwa [%]	0.05660377	0.05109489	0.04927536	0.04161412	0.09623431	...	0.25198413	0.04597701	**7.71174194**
	Ratio of EmptyRuns to Utilization Time [%]	0.432	0.439	0.441	0.435	0.447	...	0.455	0.44575	**44.308**
Target system KollRo	TransportationTime [s]	67.275	65.589	119.497	67.275	158.308	...	93.351	90.825	**98.67603**
	ScheduleDeviation [s]	3685.979	3452.219	5207.904	4637.095	5258.293	...	4829.605	1858.117	**4667.25578**
	UtilizationTime [%]	0.12	0.12	0.107	0.12	0.16	...	0.112	0.125	**11.791**
	WaitingTime [%]	0.88	0.88	0.893	0.88	0.84	...	0.888	0.875	**88.209**
	EmptyRuns [%]	1	0.98	0.959	0.964	1	...	0.941	1	**93.201**
	Input value: 900 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	805.722	676.315	1194.271	676.605	876.559	...	1079.265	872.814	**821.5058**
	ScheduleDeviation [s]	177.551	67.005	609.659	74.391	257.278	...	450.599	267.864	**218.35492**
	UtilizationTime AGV 1 [%]	0.10951009	0.03610108	0.13181818	0.42092457	0.06481481	...	0.0735786	0.07692308	**17.054785**
	UtilizationTime AGV 2 [%]	0.10086455	0.03249097	0.11818182	0.23479319	0.06944444	...	0.0735786	0.08615385	**15.3345084**
	UtilizationTime Human [%]	0.74927954	0.90252708	0.66818182	0.30656934	0.80092593	...	0.81939799	0.76307692	**61.3638043**
	UtilizationTime Kukaliwa [%]	0.04034582	0.02888087	0.08181818	0.0377129	0.06481481	...	0.03344482	0.07384615	**6.24690227**
	Ratio of EmptyRuns to Utilization Time [%]	0.437	0.46	0.45225	0.438	0.40775	...	0.43125	0.44	**46.57625**
Target system KollRo	TransportationTime [s]	186.486	156.681	117.674	43.109	117.884	...	89.925	178.18	**106.53869**
	ScheduleDeviation [s]	2133.985	3143.873	6082.306	3868.794	5878.05	...	1698.83	2010.619	**3730.41262**
	UtilizationTime [%]	0.103	0.099	0.17	0.127	0.108	...	0.125	0.109	**11.49**
	WaitingTime [%]	0.897	0.901	0.83	0.873	0.892	...	0.875	0.891	**88.51**
	EmptyRuns [%]	0.945	0.954	0.285	0.947	0.951	...	0.953	0.887	**92.73**
	Input value: 1000 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	893.191	678.099	685.848	817.936	1092.709	...	681.754	671.514	**818.9298**
	ScheduleDeviation [s]	278.053	65.441	90.485	249.99	482.902	...	84.784	98.573	**214.84334**
	UtilizationTime AGV 1 [%]	0.06	0.14666667	0.07821229	0.12292359	0.046875	...	0.20685435	0.21783439	**14.2697827**
	UtilizationTime AGV 2 [%]	0.092	0.04333333	0.12849162	0.19269103	0.134375	...	0.43451652	0.45095541	**18.4901131**
	UtilizationTime Human [%]	0.78	0.76	0.67039106	0.62126246	0.734375	...	0.29130967	0.25987261	**60.7262924**
	UtilizationTime Kukaliwa [%]	0.068	0.05	0.12290503	0.06312292	0.084375	...	0.06731946	0.07133758	**6.51381179**
	Ratio of EmptyRuns to Utilization Time [%]	0.434	0.442	0.424	0.406	0.4585	...	0.432	0.45	**47.06425**
Target system KollRo	TransportationTime [s]	65.722	122.897	43.784	120.632	87.252	...	94.928	93.351	**100.3809**
	ScheduleDeviation [s]	4188.02	2912.545	1846.372	4782.77	1546.654	...	3048.273	2921.546	**2937.44712**
	UtilizationTime [%]	0.119	0.107	0.129	0.106	0.124	...	0.113	0.112	**11.701**
	WaitingTime [%]	0.881	0.893	0.871	0.894	0.876	...	0.887	0.888	**88.299**
	EmptyRuns [%]	1	0.958	0.944	0.955	1	...	0.937	0.941	**94.636**

Appendix 18: Simulation runs of the static system for the input values between 600 and 1,000 seconds

	Input value: 100 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	2473.657	1891.323	1063.3	1643.787	1100.551	...	1772.868	1667.908	**1675.49173**
	ScheduleDeviation [s]	1827.405	1255.851	423.634	998.687	455.551	...	1098.243	1034.207	**1030.5487**
	UtilizationTime AGV 1 [%]	0.27272727	0.19153439	0.28571429	0.28277635	0.24134948	...	0.15044248	0.51136364	**27.1198509**
	UtilizationTime AGV 2 [%]	0.22960373	0.2952381	0.22425249	0.29717224	0.23269896	...	0.2300885	0.22017045	**25.678263**
	UtilizationTime Human [%]	0.39160839	0.35449735	0.33554817	0.27095116	0.31487889	...	0.56047198	0.17045455	**32.1157841**
	UtilizationTime Kukaliwa [%]	0.10606061	0.15873016	0.15448505	0.14910026	0.21107266	...	0.05899705	0.09801136	**15.086102**
	Ratio of EmptyRuns to Utilization Time [%]	0.346	0.34525	0.40375	0.30575	0.369	...	0.35875	0.376	**34.96575**
Target system KollRo	TransportationTime [s]	70.68	72.413	74.18	71.562	72.62	...	76.939	77.798	**72.60998**
	ScheduleDeviation [s]	731.964	702.647	820.507	1764.005	683.035	...	661.355	892.589	**871.83979**
	UtilizationTime [%]	0.063	0.066	0.08	0.077	0.062	...	0.153	0.148	**10.17**
	WaitingTime [%]	0.937	0.934	0.92	0.923	0.938	...	0.847	0.852	**89.811**
	EmptyRuns [%]	0.352	0.449	0.48	0.462	0.483	...	0.428	0.417	**42.981**
	Input value: 200 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	825.942	788.38	798.523	766.416	1243.663	...	859.643	809.219	**1017.50439**
	ScheduleDeviation [s]	204.681	151.59	180.39	151.471	572.607	...	204.205	174.719	**377.58794**
	UtilizationTime AGV 1 [%]	0.3645461	0.39655172	0.36954915	0.3519656	0.2953271	...	0.32653061	0.29613734	**32.4221621**
	UtilizationTime AGV 2 [%]	0.31022159	0.2966954	0.25277162	0.29914005	0.31308411	...	0.30612245	0.31974249	**29.0810448**
	UtilizationTime Human [%]	0.17012152	0.19037356	0.19364375	0.18611794	0.2046729	...	0.18859958	0.25751073	**23.1379226**
	UtilizationTime Kukaliwa [%]	0.15511079	0.11637931	0.18403548	0.16277641	0.18691589	...	0.17874736	0.12660944	**15.3588705**
	Ratio of EmptyRuns to Utilization Time [%]	0.3995	0.32925	0.3305	0.36575	0.3505	...	0.3565	0.354	**36.08325**
Target system KollRo	TransportationTime [s]	69.406	71.796	72.108	71.378	70.14	...	72.872	71.236	**72.29378**
	ScheduleDeviation [s]	1689.801	1794.875	1681.587	909.015	983.259	...	1754.879	1598.884	**1339.6702**
	UtilizationTime [%]	0.085	0.097	0.093	0.102	0.044	...	0.095	0.092	**10.776**
	WaitingTime [%]	0.915	0.903	0.907	0.898	0.956	...	0.905	0.908	**89.187**
	EmptyRuns [%]	0.463	0.456	0.465	0.437	0.459	...	0.409	0.46	**44.494**
	Input value: 300 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	945.8	1301.054	916.848	912.044	881.246	...	909.117	944.544	**968.45454**
	ScheduleDeviation [s]	303.768	666.511	273.67	271.321	255.334	...	258.675	336.691	**338.27797**
	UtilizationTime AGV 1 [%]	0.37	0.36788049	0.32844037	0.3572068	0.32894737	...	0.35273369	0.43015726	**34.5523647**
	UtilizationTime AGV 2 [%]	0.31636364	0.3155929	0.35963303	0.32587287	0.35350877	...	0.31922399	0.24606846	**32.9513343**
	UtilizationTime Human [%]	0.20090909	0.1774043	0.17889908	0.17547001	0.16578947	...	0.18959436	0.18963922	**18.8324633**
	UtilizationTime Kukaliwa [%]	0.11272727	0.13912232	0.13302752	0.14145031	0.15175439	...	0.13844797	0.13413506	**13.6638376**
	Ratio of EmptyRuns to Utilization Time [%]	0.384	0.385	0.3965	0.38975	0.3855	...	0.39625	0.396	**37.316**
Target system KollRo	TransportationTime [s]	69.458	70.67	72.841	68.98	73.971	...	74.488	70.167	**72.31103**
	ScheduleDeviation [s]	940.194	1025.738	899.327	1038.967	954.792	...	945.478	1025.629	**941.64452**
	UtilizationTime [%]	0.084	0.082	0.09	0.061	0.09	...	0.088	0.075	**8.576**
	WaitingTime [%]	0.916	0.918	0.91	0.939	0.91	...	0.912	0.925	**91.424**
	EmptyRuns [%]	0.424	0.444	0.462	0.465	0.473	...	0.444	0.448	**44.816**
	Input value: 400 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	712.985	1336.495	734.025	968.871	949.732	...	726.635	718.693	**901.06283**
	ScheduleDeviation [s]	115.215	712.417	107.872	327.103	325.11	...	122.174	79.041	**273.02176**
	UtilizationTime AGV 1 [%]	0.43176471	0.45505618	0.29129464	0.33114035	0.34002361	...	0.40614709	0.46584546	**37.0772268**
	UtilizationTime AGV 2 [%]	0.33529412	0.20337079	0.36495536	0.35635965	0.4120425	...	0.31613611	0.29675252	**35.1893662**
	UtilizationTime Human [%]	0.12823529	0.20337079	0.21205357	0.17982456	0.14285714	...	0.13830955	0.11982083	**15.0248624**
	UtilizationTime Kukaliwa [%]	0.10470588	0.13820225	0.13169643	0.13267544	0.10507674	...	0.13940724	0.11758119	**12.7085446**
	Ratio of EmptyRuns to Utilization Time [%]	0.38525	0.387	0.39675	0.386	0.37375	...	0.3865	0.409	**37.88225**
Target system KollRo	TransportationTime [s]	72.991	73.557	71.574	73.735	72.041	...	72.34	72.4	**72.79389**
	ScheduleDeviation [s]	803.264	862.36	755.384	892.155	867.742	...	951.675	812.019	**851.14102**
	UtilizationTime [%]	0.075	0.078	0.061	0.078	0.068	...	0.066	0.071	**7.33**
	WaitingTime [%]	0.925	0.922	0.939	0.922	0.932	...	0.934	0.929	**92.667**
	EmptyRuns [%]	0.5	0.463	0.418	0.442	0.463	...	0.498	0.458	**45.748**
	Input value: 500 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	981.047	1073.484	857.461	836.663	1157.829	...	725.124	863.075	**873.36778**
	ScheduleDeviation [s]	342.672	468.067	238.649	184.766	524.715	...	107.519	248.296	**243.86759**
	UtilizationTime AGV 1 [%]	0.4587766	0.31546135	0.40310078	0.46842105	0.29949239	...	0.45034014	0.30120482	**39.3097244**
	UtilizationTime AGV 2 [%]	0.33111702	0.40024938	0.32816537	0.27763158	0.43527919	...	0.30204082	0.43641232	**36.0492782**
	UtilizationTime Human [%]	0.10904255	0.14837905	0.13565891	0.125	0.14720812	...	0.14421769	0.145917	**13.7086367**
	UtilizationTime Kukaliwa [%]	0.10106383	0.13591022	0.13307494	0.12894737	0.1180203	...	0.10340136	0.11646586	**10.9323607**
	Ratio of EmptyRuns to Utilization Time [%]	0.4185	0.38425	0.40925	0.40375	0.408	...	0.40575	0.39725	**38.79575**
Target system KollRo	TransportationTime [s]	71.432	69.162	69.736	72.305	71.058	...	71.432	73.218	**71.5996**
	ScheduleDeviation [s]	784.594	745.406	884.305	778.406	872.705	...	761.898	782.869	**811.87927**
	UtilizationTime [%]	0.062	0.064	0.06	0.062	0.062	...	0.059	0.064	**6.197**
	WaitingTime [%]	0.938	0.936	0.94	0.938	0.938	...	0.941	0.936	**93.803**
	EmptyRuns [%]	0.457	0.421	0.488	0.452	0.505	...	0.497	0.481	**47.461**

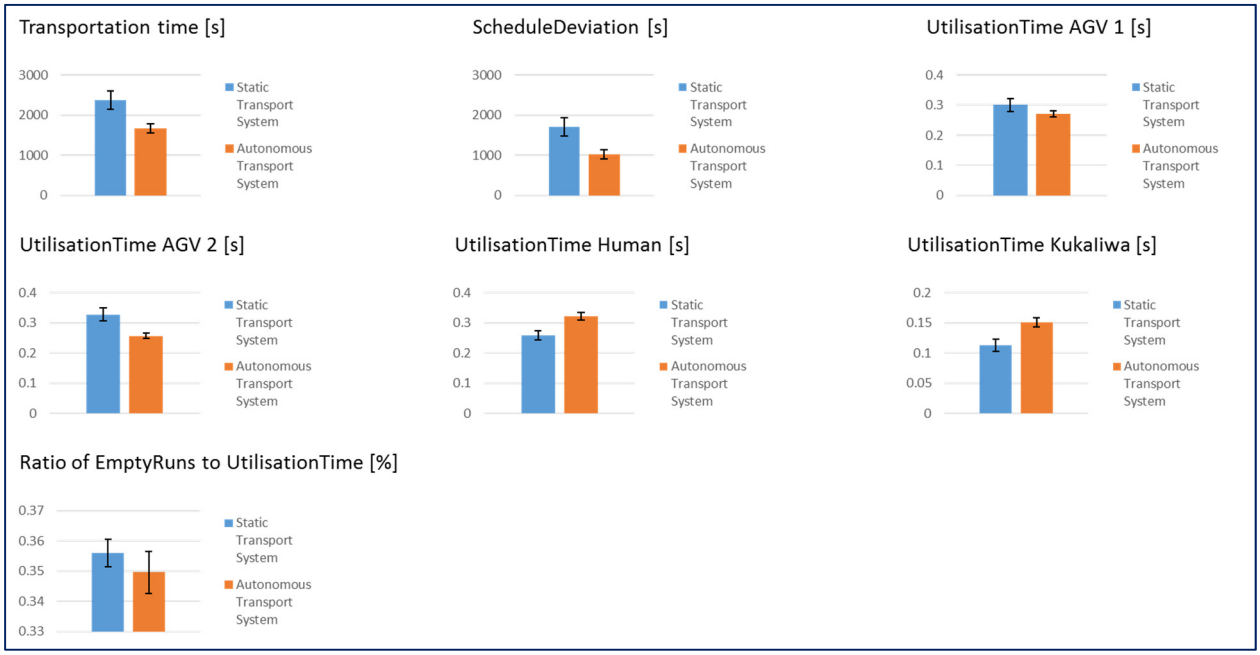
Appendix 19: Simulation runs of the autonomous system for the input values between 100 and 500 seconds

	Input value: 600 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	710.26	862.259	837.048	873.973	697.14	...	927.724	929.473	**848.07987**
	ScheduleDeviation [s]	77.154	219.156	234.256	237.906	113.449	...	318.565	304.132	**230.48188**
	UtilizationTime AGV 1 [%]	0.38729198	0.38554217	0.38586156	0.35204856	0.40685544	...	0.36581709	0.39652449	**39.4779152**
	UtilizationTime AGV 2 [%]	0.40393343	0.39156627	0.39322533	0.37936267	0.35022355	...	0.4017991	0.3728278	**37.2440539**
	UtilizationTime Human [%]	0.10741301	0.125	0.13402062	0.14415781	0.15946349	...	0.11994003	0.11690363	**12.9744075**
	UtilizationTime Kukaliwa [%]	0.10136157	0.09789157	0.08689249	0.12443096	0.08345753	...	0.11244378	0.11374408	**10.3036235**
	Ratio of EmptyRuns to Utilization Time [%]	0.39275	0.41075	0.4245	0.3935	0.41475	...	0.39275	0.39925	**39.44525**
Target system KollRo	TransportationTime [s]	75.392	73.382	72.693	72.709	72.156	...	70.722	71.204	**73.61904**
	ScheduleDeviation [s]	806.521	809.904	889.607	916.22	979.777	...	936.082	838.913	**861.6406**
	UtilizationTime [%]	0.055	0.054	0.053	0.048	0.05	...	0.05	0.055	**5.391**
	WaitingTime [%]	0.945	0.946	0.947	0.952	0.95	...	0.95	0.945	**94.608**
	EmptyRuns [%]	0.445	0.432	0.464	0.449	0.465	...	0.447	0.478	**45.061**
	Input value: 700 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	710.756	800.408	970.249	789.119	693.17	...	844.371	703.834	**835.41577**
	ScheduleDeviation [s]	73.648	167.729	354.79	146.545	91.871	...	263.102	79.063	**219.46866**
	UtilizationTime AGV 1 [%]	0.52380952	0.40798611	0.36777583	0.27863248	0.51290878	...	0.42367067	0.4359401	**40.7328465**
	UtilizationTime AGV 2 [%]	0.27750411	0.37152778	0.43082312	0.4991453	0.27022375	...	0.3567753	0.35607321	**37.5849916**
	UtilizationTime Human [%]	0.11001642	0.13541667	0.10157618	0.11623932	0.13080895	...	0.12178388	0.10648918	**12.3816714**
	UtilizationTime Kukaliwa [%]	0.08866995	0.08506944	0.09982487	0.10598291	0.08605852	...	0.09777015	0.1014975	**9.30049055**
	Ratio of EmptyRuns to Utilization Time [%]	0.41125	0.40875	0.4065	0.43725	0.42425	...	0.416	0.41325	**39.60175**
Target system KollRo	TransportationTime [s]	77.928	75.206	72.644	78.084	75.662	...	73.027	75.779	**75.68269**
	ScheduleDeviation [s]	826.225	1006.221	832.745	992.864	900.428	...	663.944	681.364	**872.49577**
	UtilizationTime [%]	0.047	0.048	0.049	0.041	0.045	...	0.053	0.046	**4.806**
	WaitingTime [%]	0.949	0.952	0.951	0.959	0.955	...	0.947	0.954	**95.167**
	EmptyRuns [%]	0.377	0.452	0.415	0.432	0.451	...	0.464	0.416	**44.313**
	Input value: 800 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	710.083	690.302	1173.342	815.765	967.95	...	830.746	884.04	**829.14631**
	ScheduleDeviation [s]	94.644	81.914	563.554	204.441	362.544	...	208.096	277.246	**211.537**
	UtilizationTime AGV 1 [%]	0.35984848	0.39809524	0.30097087	0.40697674	0.46303502	...	0.67910448	0.38505747	**40.8480966**
	UtilizationTime AGV 2 [%]	0.47159091	0.40380952	0.49708738	0.38178295	0.3229572	...	0.15422886	0.39463602	**38.103069**
	UtilizationTime Human [%]	0.08901515	0.11047619	0.10291262	0.10465116	0.12256809	...	0.10945274	0.12260536	**11.8688869**
	UtilizationTime Kukaliwa [%]	0.07954545	0.08761905	0.09902913	0.10658915	0.09143969	...	0.05721393	0.09770115	**9.17994748**
	Ratio of EmptyRuns to Utilization Time [%]	0.41675	0.426	0.401	0.416	0.4235	...	0.436	0.427	**39.802**
Target system KollRo	TransportationTime [s]	77.041	76.342	74.338	75.417	76.078	...	79.028	73.029	**75.86277**
	ScheduleDeviation [s]	855.608	956.238	1042.93	877.045	832.726	...	650.974	816.111	**892.61634**
	UtilizationTime [%]	0.04	0.042	0.052	0.042	0.046	...	0.063	0.046	**4.517**
	WaitingTime [%]	0.96	0.958	0.948	0.958	0.954	...	0.937	0.954	**95.476**
	EmptyRuns [%]	0.425	0.477	0.432	0.453	0.438	...	0.405	0.438	**43.631**
	Input value: 900 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	950.318	815.024	939.652	827.551	1038.855	...	697.271	711.775	**824.63091**
	ScheduleDeviation [s]	320.003	218.511	286.952	211.898	399.614	...	79.285	78.904	**207.92101**
	UtilizationTime AGV 1 [%]	0.42116631	0.50204918	0.40748899	0.47608696	0.58093126	...	0.44273128	0.45061728	**42.6927286**
	UtilizationTime AGV 2 [%]	0.32613391	0.30737705	0.42070485	0.34347826	0.23503326	...	0.37444934	0.35390947	**36.8229678**
	UtilizationTime Human [%]	0.15982721	0.11885246	0.1123348	0.1173913	0.11086475	...	0.1123348	0.10493827	**12.0901321**
	UtilizationTime Kukaliwa [%]	0.09287257	0.07172131	0.05947137	0.06304348	0.07317073	...	0.07048458	0.09053498	**8.39417147**
	Ratio of EmptyRuns to Utilization Time [%]	0.41475	0.43525	0.4135	0.4315	0.42925	...	0.40275	0.416	**40.119**
Target system KollRo	TransportationTime [s]	76.046	78.667	78.699	77.597	78.667	...	78.667	77.745	**76.70913**
	ScheduleDeviation [s]	874.63	859.708	1027.486	938.134	892.588	...	909.262	847.674	**892.22726**
	UtilizationTime [%]	0.042	0.045	0.045	0.044	0.043	...	0.045	0.046	**4.316**
	WaitingTime [%]	0.958	0.955	0.955	0.956	0.957	...	0.955	0.954	**95.684**
	EmptyRuns [%]	0.431	0.431	0.438	0.431	0.438	...	0.453	0.476	**43.937**
	Input value: 1000 [s]	Run 1	Run 2	Run 3	Run4	Run 5	...	Run 99	Run 100	Average
Target system Means of transport	TransportationTime [s]	780.854	1104.643	715.879	828.11	685.38	...	707.398	1130.507	**820.6179**
	ScheduleDeviation [s]	185.923	477.348	84.886	229.026	97.86	...	102.047	489.92	**207.00604**
	UtilizationTime AGV 1 [%]	0.44180523	0.36384439	0.35251799	0.48470588	0.44217687	...	0.34953704	0.33886256	**40.2908129**
	UtilizationTime AGV 2 [%]	0.33966746	0.40503432	0.42206235	0.33176471	0.36961451	...	0.46296296	0.39099526	**38.6960105**
	UtilizationTime Human [%]	0.15914489	0.14416476	0.12470024	0.10117647	0.0952381	...	0.10185185	0.18246445	**12.5373424**
	UtilizationTime Kukaliwa [%]	0.05938242	0.08695652	0.10071942	0.08235294	0.09297052	...	0.08564815	0.08767773	**8.47583428**
	Ratio of EmptyRuns to Utilization Time [%]	0.42125	0.4255	0.45825	0.435	0.45425	...	0.43925	0.439	**40.8065**
Target system KollRo	TransportationTime [s]	74.832	75.794	75.26	77.454	78.664	...	75.976	70.837	**76.15827**
	ScheduleDeviation [s]	997.703	837.891	686.12	626.315	789.068	...	633.135	1005.259	**844.66686**
	UtilizationTime [%]	0.038	0.038	0.037	0.043	0.04	...	0.042	0.038	**3.894**
	WaitingTime [%]	0.962	0.962	0.956	0.957	0.96	...	0.958	0.962	**96.075**
	EmptyRuns [%]	0.413	0.413	0.432	0.432	0.433	...	0.433	0.456	**44.068**

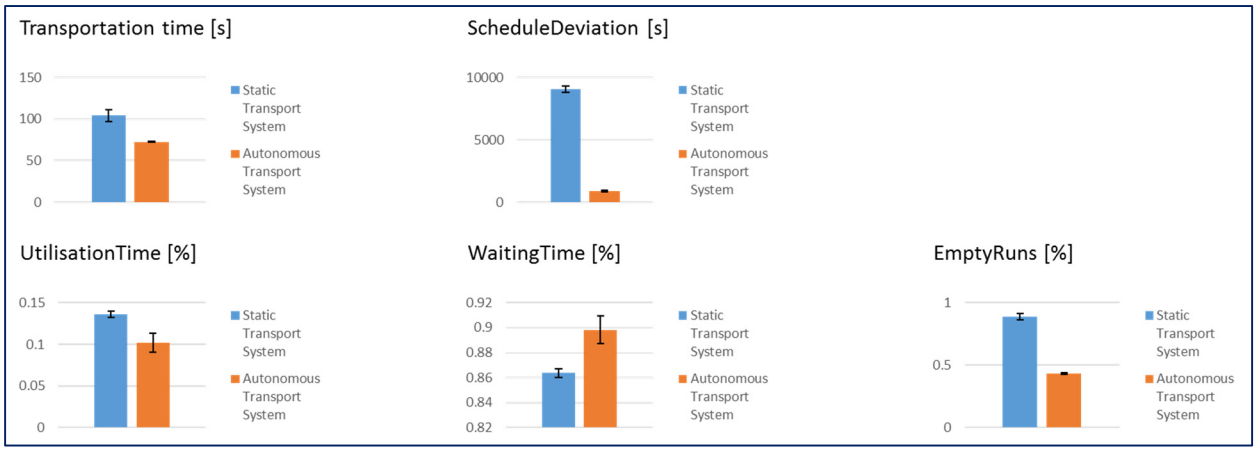
Appendix 20: Simulation runs of the autonomous system for the input values between 600 and 1,000 seconds

E.3 Confidence intervals of the simulation data

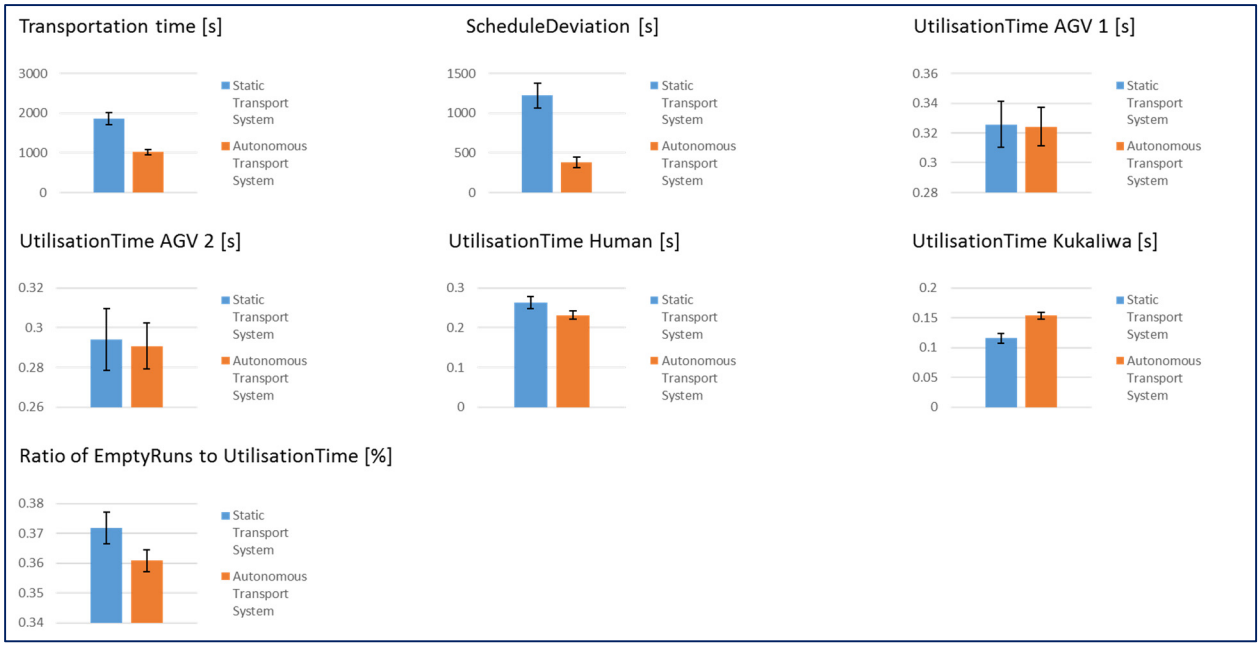
For this work, confidence intervals for the target criteria were calculated in order to analyse the relationship among themselves. In the following, the confidence intervals of the target system of the means of transport and the target system of the KollRo are shown for each input variable



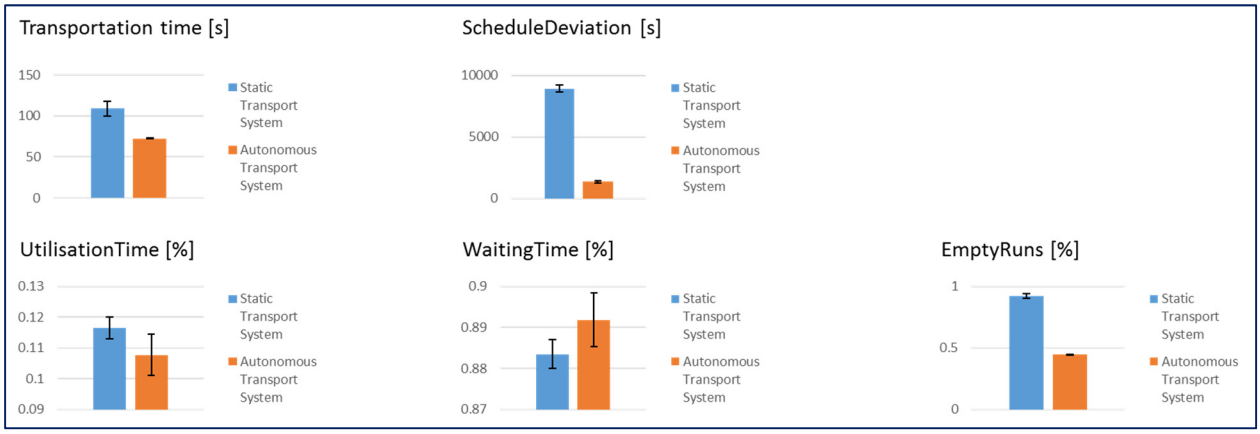
Appendix 21: Confidence intervals of the target system of the means of transport at an input variable of 100 seconds



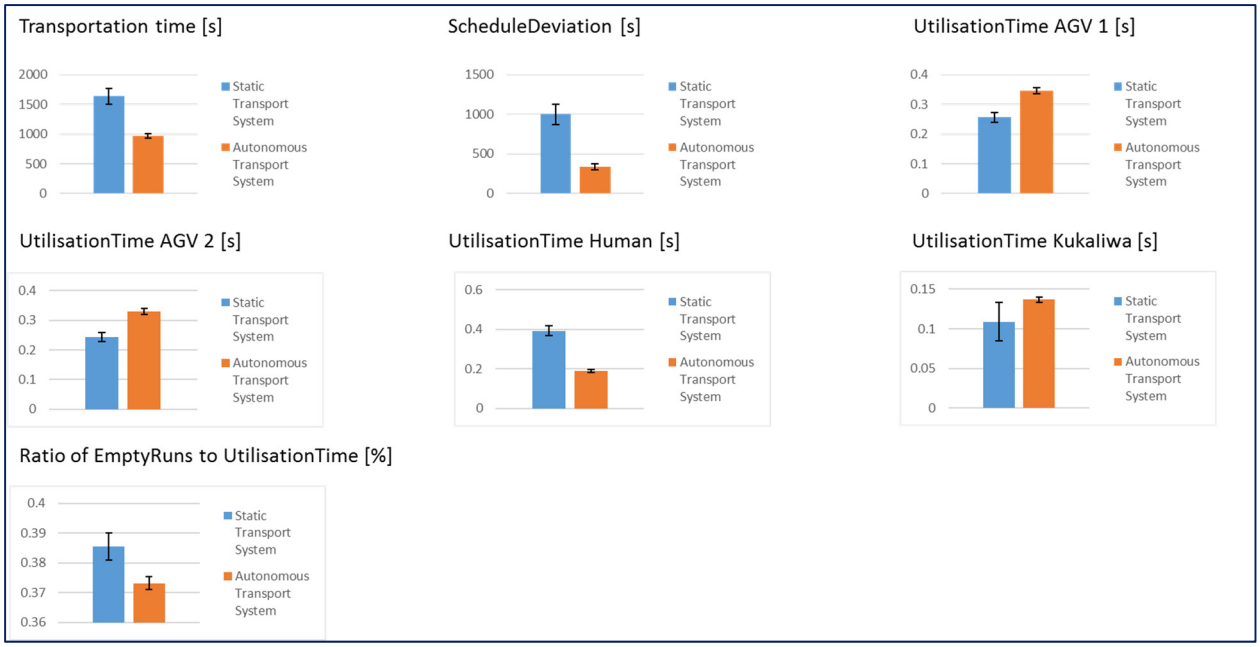
Appendix 22: Confidence intervals of the target system of the KollRo at an input variable of 100 seconds



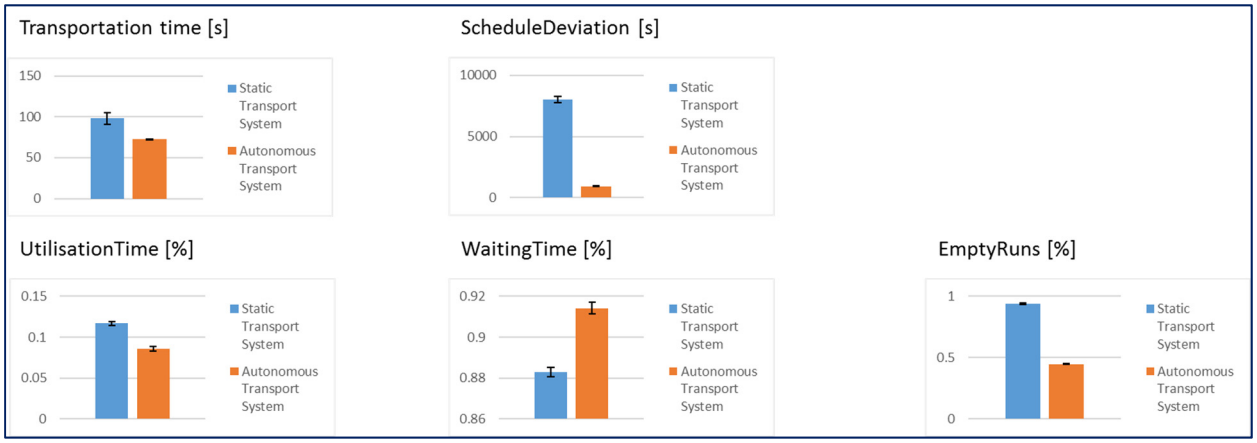
Appendix 23: Confidence intervals of the target system of the means of transport at an input variable of 200 seconds



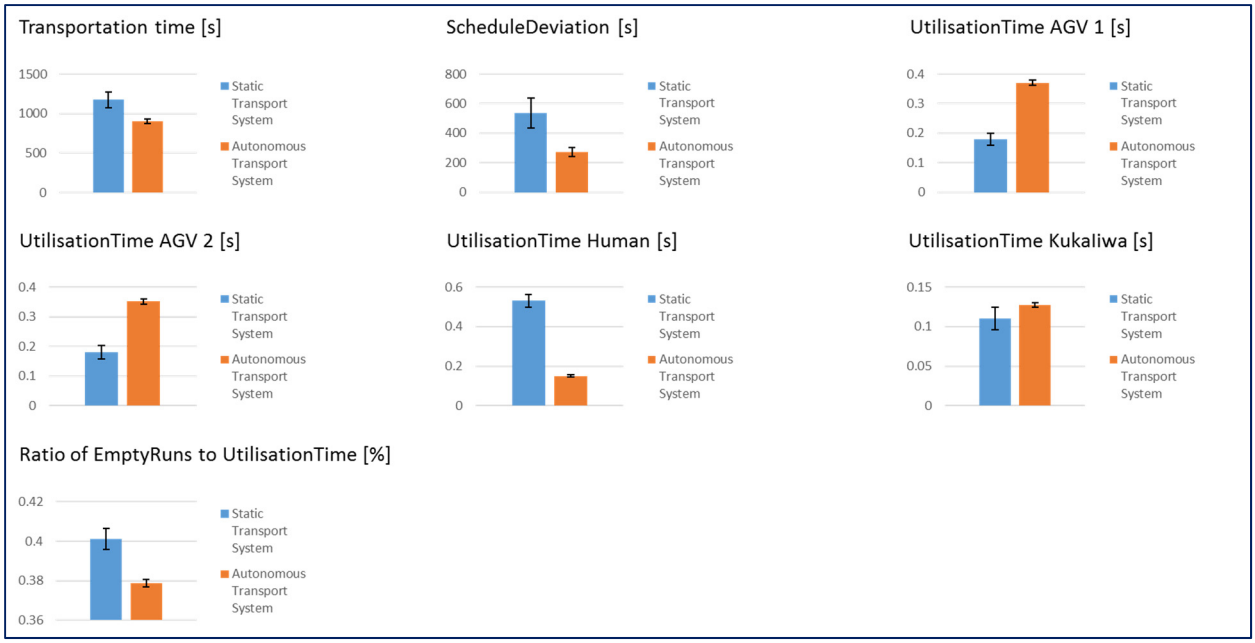
Appendix 24: Confidence intervals of the target system of the KollRo at an input variable of 200 seconds



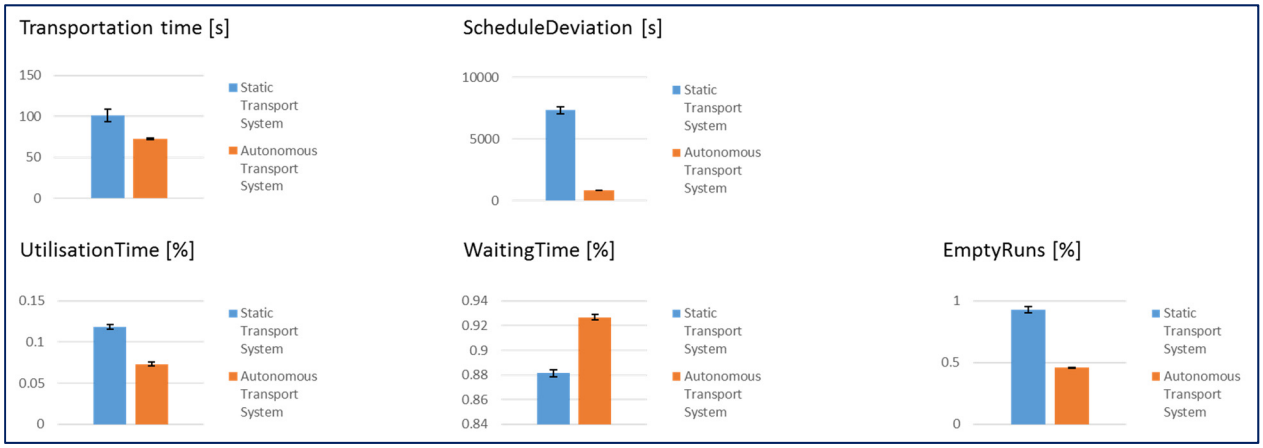
Appendix 25: Confidence intervals of the target system of the means of transport at an input variable of 300 seconds



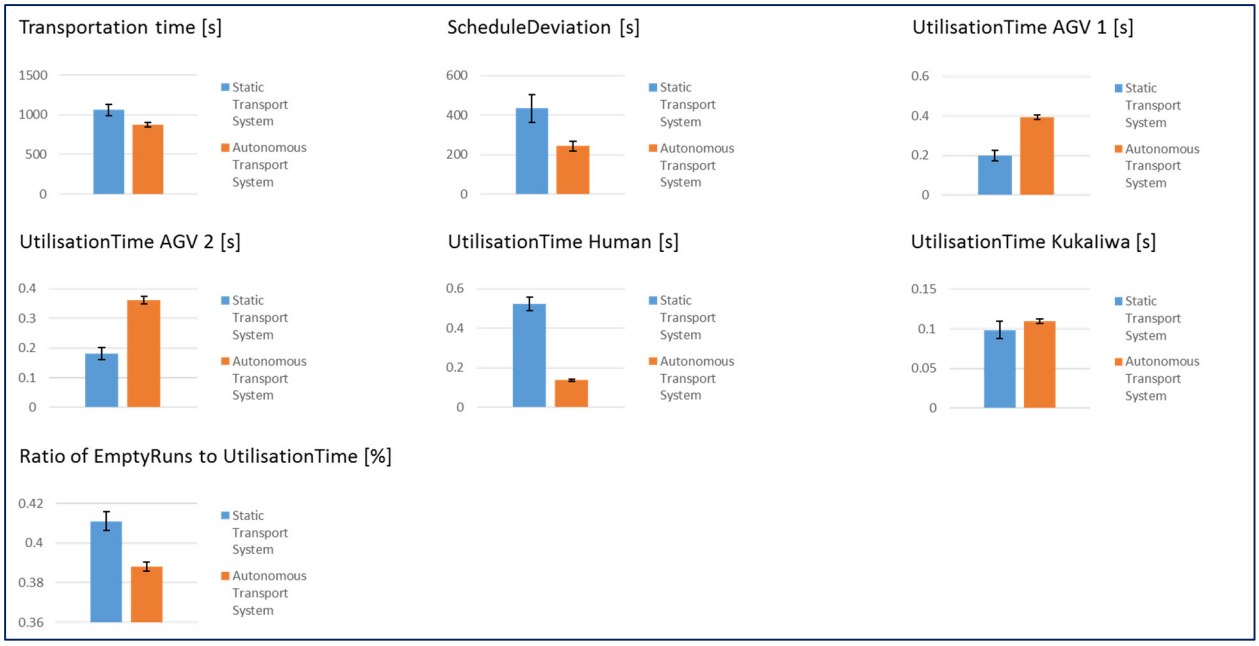
Appendix 26: Confidence intervals of the target system of the KollRo at an input variable of 300 seconds



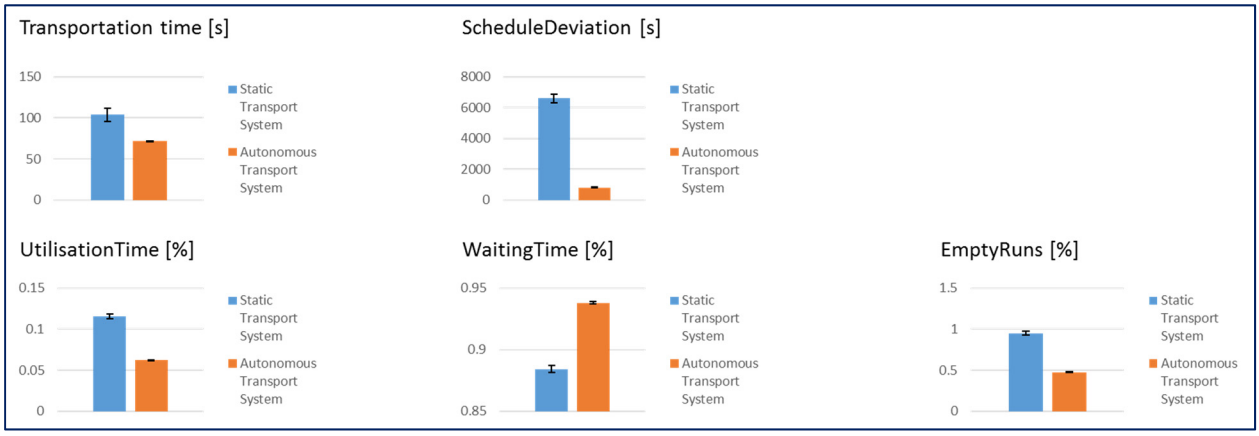
Appendix 27: Confidence intervals of the target system of the means of transport at an input variable of 400 seconds



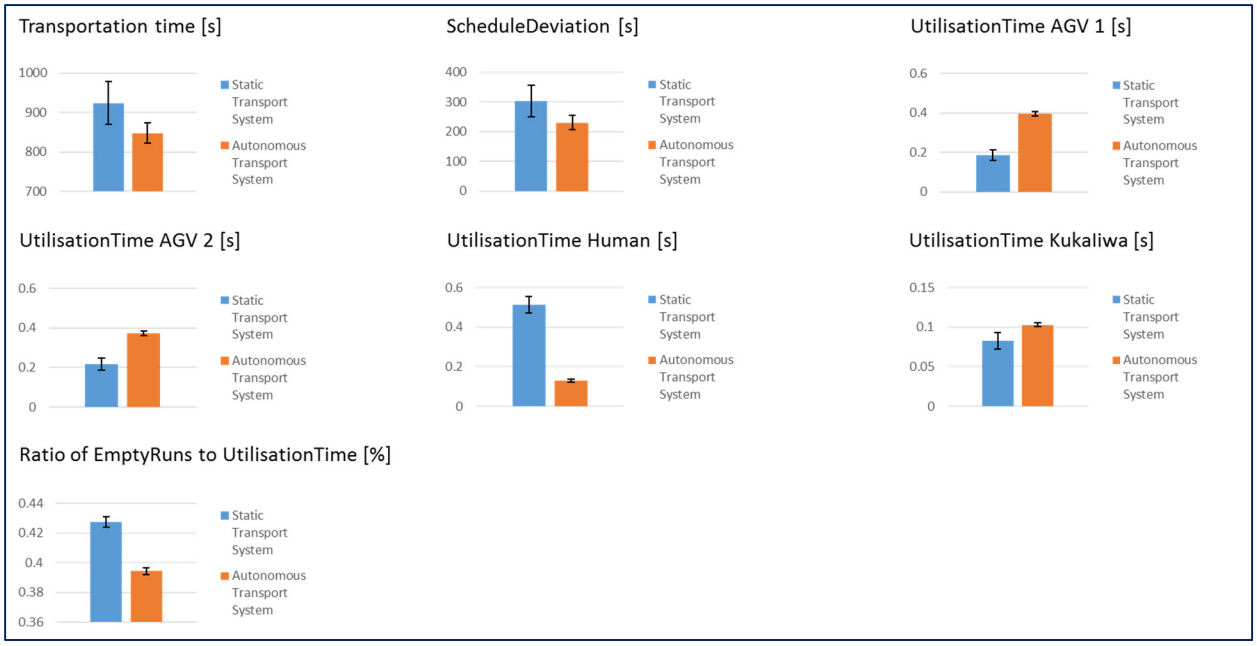
Appendix 28: Confidence intervals of the target system of the KollRo at an input variable of 400 seconds



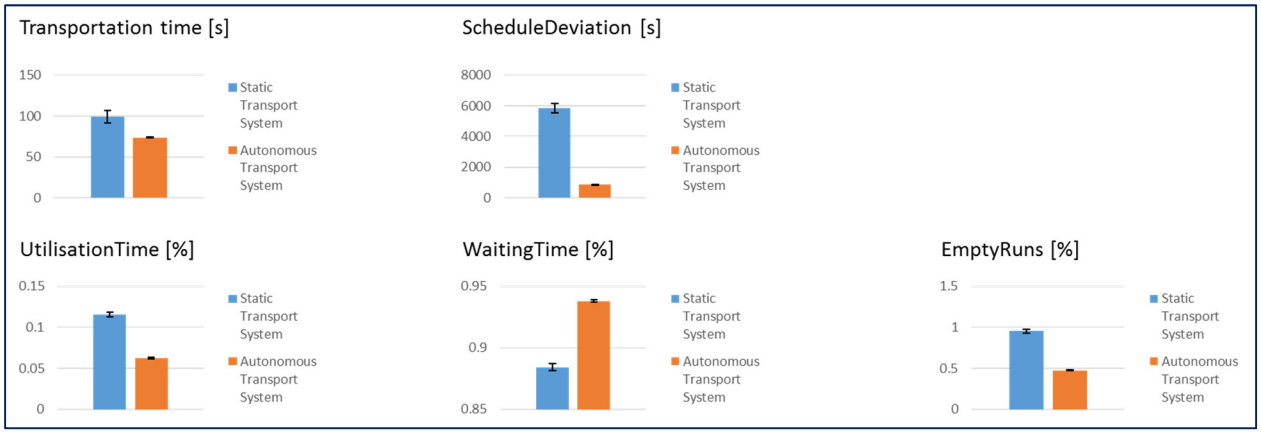
Appendix 29: Confidence intervals of the target system of the means of transport at an input variable of 500 seconds



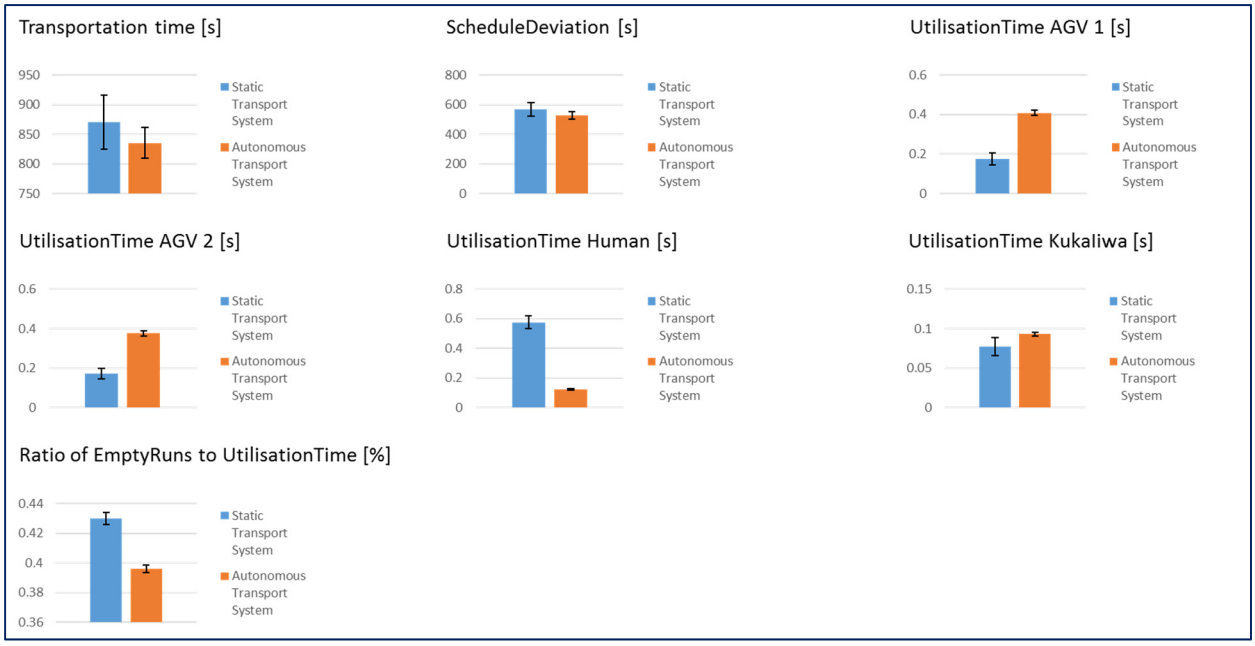
Appendix 30: Confidence intervals of the target system of the KollRo at an input variable of 500 seconds



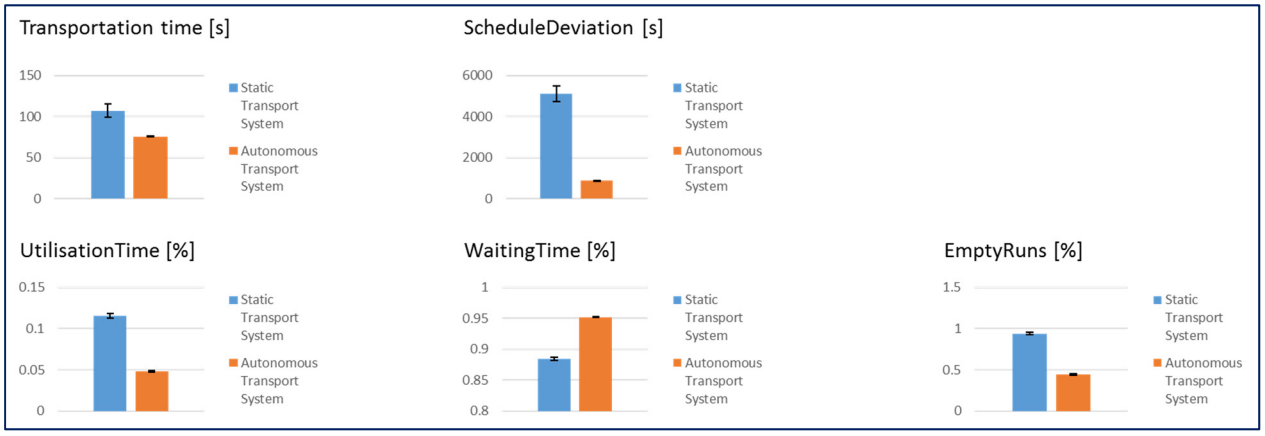
Appendix 31: Confidence intervals of the target system of the means of transport at an input variable of 600 seconds



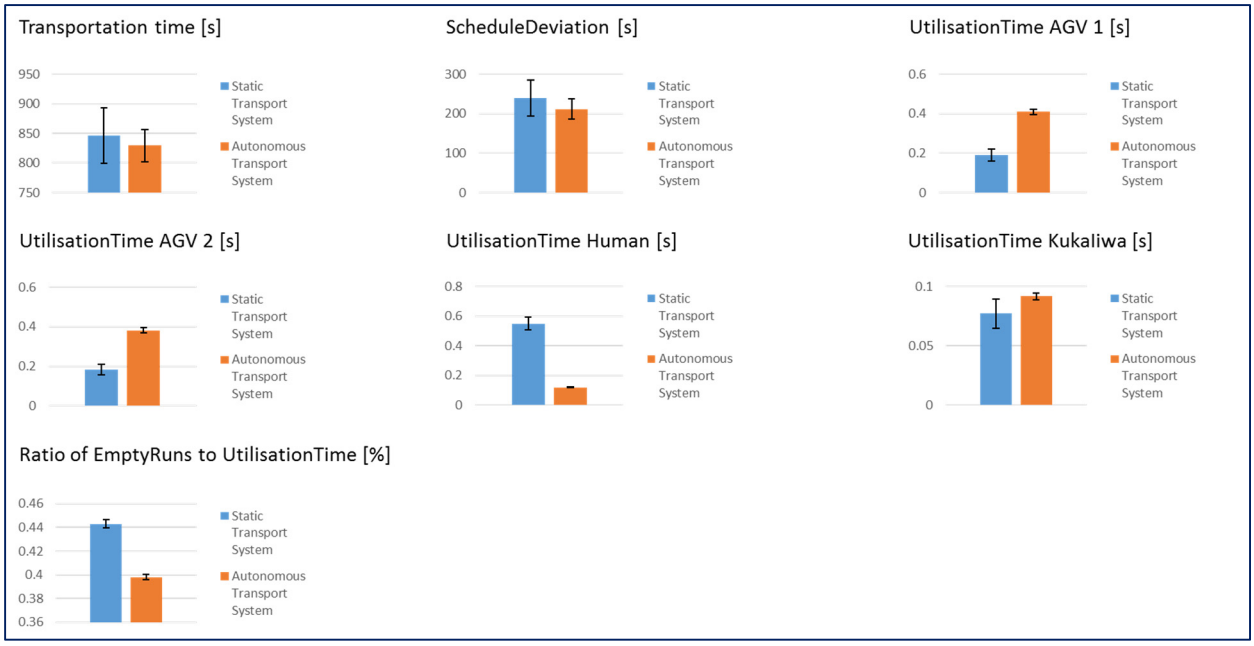
Appendix 32: Confidence intervals of the target system of the KollRo at an input variable of 600 seconds



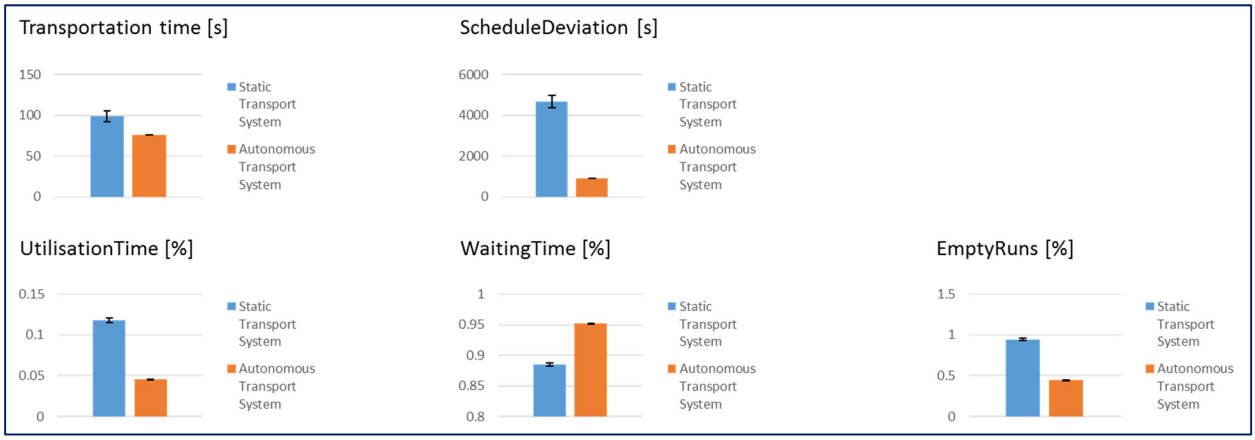
Appendix 33: Confidence intervals of the target system of the means of transport at an input variable of 700 seconds



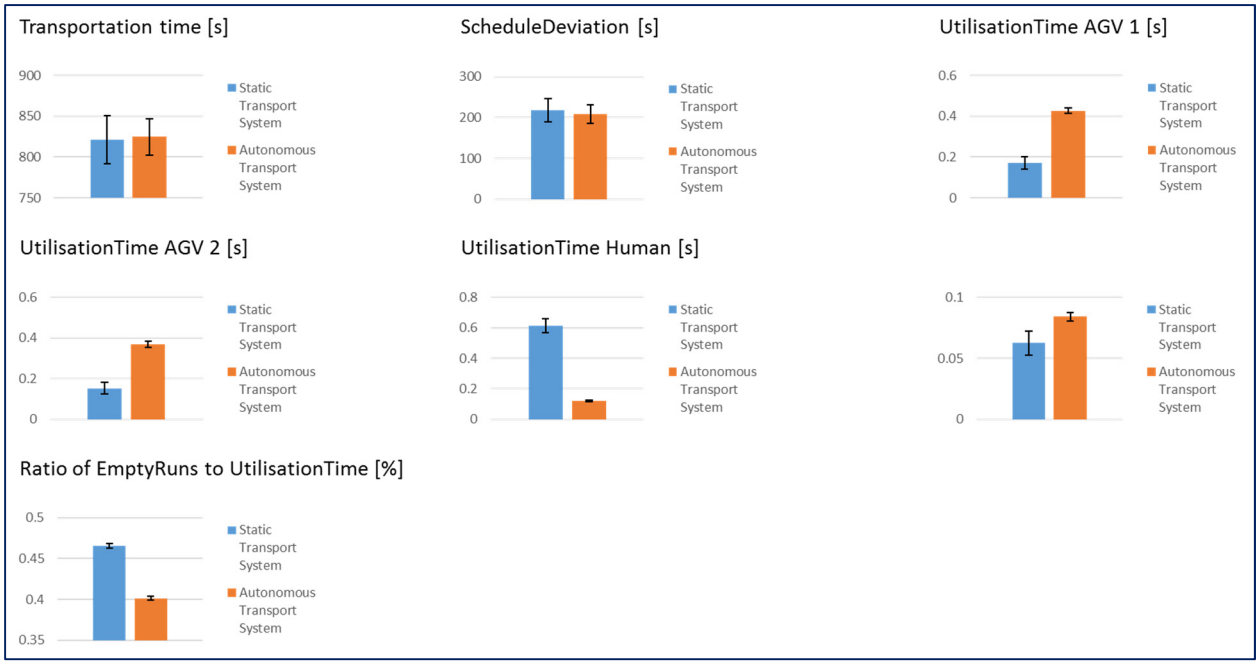
Appendix 34: Confidence intervals of the target system of the KollRo at an input variable of 700 seconds



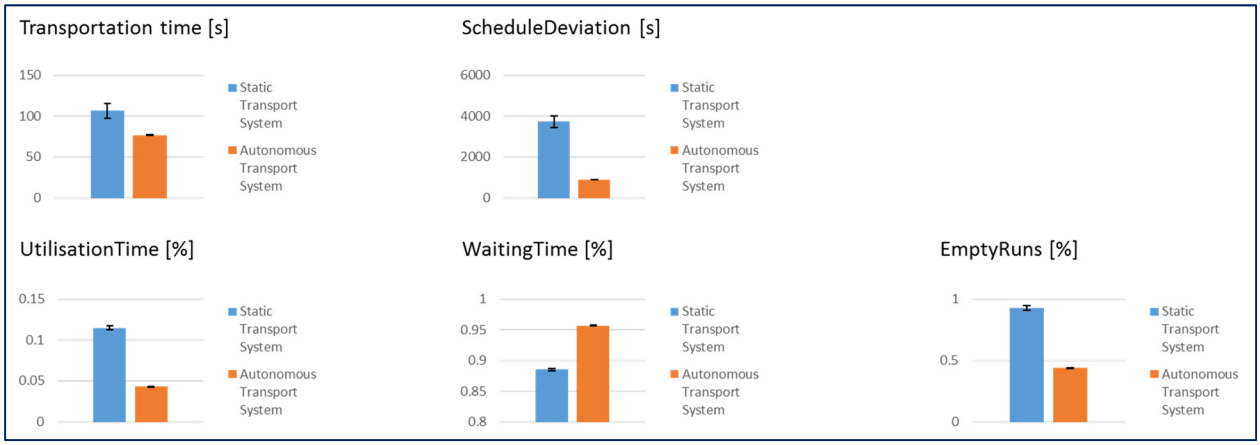
Appendix 35: Confidence intervals of the target system of the means of transport at an input variable of 800 seconds



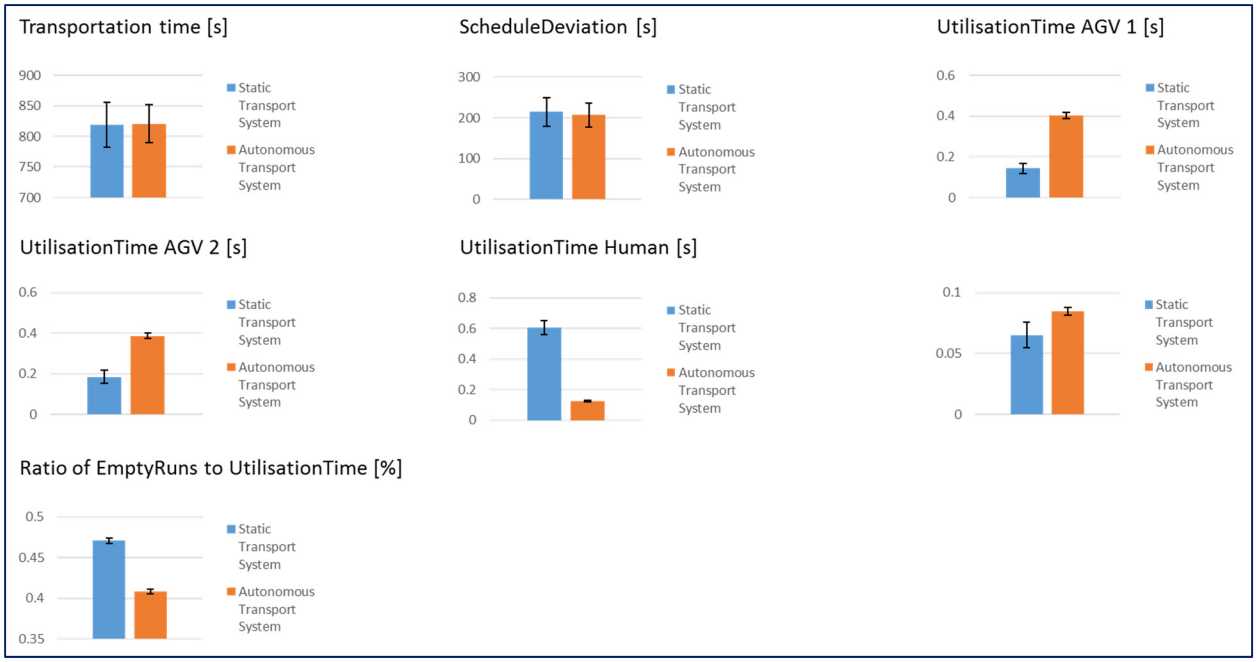
Appendix 36: Confidence intervals of the target system of the KollRo at an input variable of 800 seconds



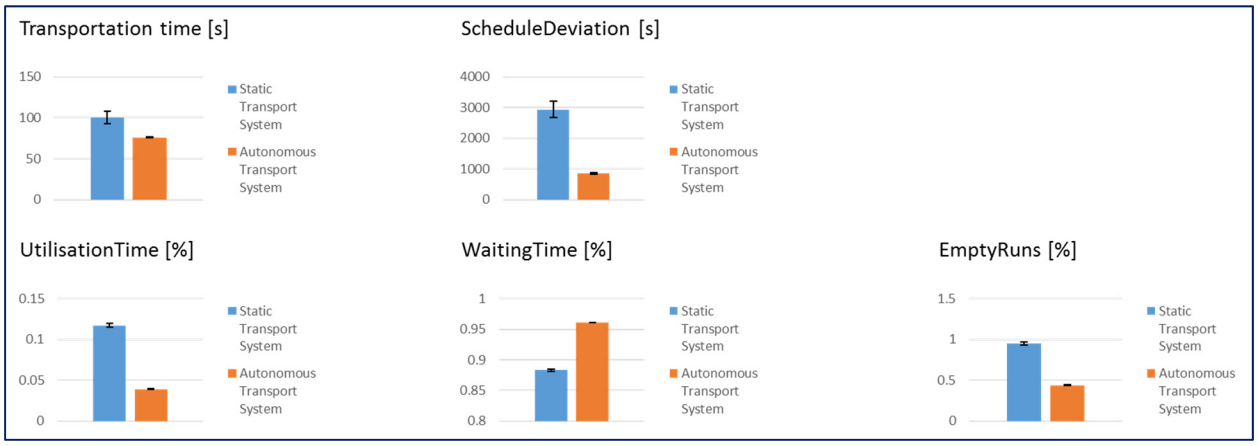
Appendix 37: Confidence intervals of the target system of the means of transport at an input variable of 900 seconds



Appendix 38: Confidence intervals of the target system of the KollRo at an input variable of 900 seconds



Appendix 39: Confidence intervals of the target system of the means of transport at an input variable of 1,000 seconds



Appendix 40: Confidence intervals of the target system of the means of transport at an input variable of 1,000 seconds

E.4 Data of the validation in the Werk150

The obtained data during the validation are stored in a database to compare them with the data from the simulation. Therefore, the data of 100 transport orders were stored.

		Simulation study	LFW150	Deviation [%]
1	TaskID	1	1	False
1	Source	1	1	False
1	Sink	5	5	False
1	Means of transport	Kukaliwa	Kukaliwa	False
1	x-Position [m]	6.511194653	6.379290886	2.07
1	y-Position [m]	20.32665943	20.04214629	1.42
1	Utilisation time [s]	79.12919822	83	4.66
1	Empty runs [s]	15.64790217	17	7.95
2	TaskID	2	2	False
2	Source	11	11	False
2	Sink	2	2	False
2	Means of transport	Human	Human	False
2	x-Position [m]	21.75	21.64462898	0.49
2	y-Position [m]	6.333333333	6.175467143	2.56
2	Utilisation time [s]	95.65375365	94	1.76
2	Empty runs [s]	77.83136148	74	5.18
3	TaskID	3	3	False
3	Source	4	4	False
3	Sink	1	1	False
3	Means of transport	Kukaliwa	Kukaliwa	False
3	x-Position [m]	20	20.13784915	0.68
3	y-Position [m]	10.16666667	10.09142662	0.75
3	Utilisation time [s]	80.97515043	86	5.84
3	Empty runs [s]	82.56042629	76	8.63
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
97	TaskID	97	97	False
97	Source	7	7	False
97	Sink	6	6	False
97	Means of transport	Kukaliwa	Kukaliwa	False
97	x-Position [m]	22.70833333	22.43455639	1.22
97	y-Position [m]	5	5.12927915	2.52
97	Utilisation time [s]	79.98463778	75	6.65
97	Empty runs [s]	15.27777778	16	4.51
98	TaskID	98	98	False
98	Source	6	6	False
98	Sink	15	15	False
98	Means of transport	KollRo	KollRo	False
98	x-Position [m]	22.66666667	22.55558822	0.49
98	y-Position [m]	5	5.135698003	2.64
98	Utilisation time [s]	58.66760905	56	4.76
98	Empty runs [s]	23.91443092	23	3.98
99	TaskID	99	99	False
99	Source	6	6	False
99	Sink	1	1	False
99	Means of transport	Kukaliwa	Kukaliwa	False
99	x-Position [m]	21.75	21.72645715	0.11
99	y-Position [m]	6.333333333	6.387049595	0.84
99	Utilisation time [s]	3.363485533	3.291666667	2.18
99	Empty runs [s]	39.34444444	37	6.34
100	TaskID	100	100	False
100	Source	15	15	False
100	Sink	1	1	False
100	Means of transport	KollRo	KollRo	False
100	x-Position [m]	24.83333333	25.0587819	0.90
100	y-Position [m]	11.58333333	11.79784532	1.82
100	Utilisation time [s]	113.703825	116	1.98
100	Empty runs [s]	6.317394111	6	5.29

	TaskID	Source	Sink	Means of transport	x-Position [m]	y-Position [m]	Utilisation time [%]	Empty runs [%]
Mean Deviation	False	False	False	False	0.24	0.17	24.66	4.80

Appendix 41: Data of the validation in the Werk150

E.5 Paired Student's t test of the data of the validation

For the validation of the simulation study, the data of the x- and y-position, the utilisation time and the time for empty runs of the means of transport were compared. To prove that there are no statistically significant differences between the data of the simulation and the obtained data from the Werk150, a paired Student's t test for each data set was conducted. The test was performed at a significance level of $\alpha = 0.05$. In the following, the results of the paired Student's t test for the data of the x- and y-position, the utilisation time and time for empty runs are shown. For all four data sets the value 't Stat' is below the value of 't Critical' for the two tail test, thus the data of the simulation and the data of Werk150 are statistically equal.

	Simulation	*Werk150*
Mean	14.8396872	14.7695437
Variance	58.0724565	58.3867944
Observations	100	100
Pearson Correlation	0.9995068	
Hypothesized Mean Difference	0	
df	99	
t Stat	1.25312955	
P(T<=t) one-tail	0.10655509	
t Critical one-tail	1.66039116	
P(T<=t) two-tail	0.21311018	
t Critical two-tail	1.98421695	

Appendix 42: Paired Student's t test for the x-position of a means of transport

	Simulation	*Werk150*
Mean	13.6803307	13.7001887
Variance	26.5006387	26.6434106
Observations	100	100
Pearson Correlation	0.99808961	
Hypothesized Mean Difference	0	
df	99	
t Stat	-1.8768269	
P(T<=t) one-tail	0.03174331	
t Critical one-tail	1.66039116	
P(T<=t) two-tail	0.06348661	
t Critical two-tail	1.98421695	

Appendix 43: Paired Student's t test for the y-position of a means of transport

	Simulation	*Werk150*
Mean	81.7803647	81.5658333
Variance	356.832876	363.154222
Observations	100	100
Pearson Correlation	0.97633817	
Hypothesized Mean Difference	0	
df	99	
t Stat	0.39830621	
P(T<=t) one-tail	0.34563109	
t Critical one-tail	1.66039116	
P(T<=t) two-tail	0.69126218	
t Critical two-tail	1.98421695	

Appendix 44: Paired Student's t test for the utilisation time of a means of transport

	Simulation	*Werk150*
Mean	169.724134	170.76
Variance	203851.904	208720.548
Observations	100	100
Pearson Correlation	0.9998137	
Hypothesized Mean Difference	0	
df	99	
t Stat	-1.0567535	
P(T<=t) one-tail	0.14659801	
t Critical one-tail	1.66039116	
P(T<=t) two-tail	0.29319601	
t Critical two-tail	1.98421695	

Appendix 45: Paired Student's t test for the time of empty runs of a means of transport